Processing チートシート

Ver. 1.0

Processing 3.0　Written by NISHIDA Ryota　http://dev.eyln.com

基本

▲

```
// はじめの準備（2D）
void setup() {
  size(800, 600, P2D);
}

// 毎フレームすること
void draw() {
}
```

```
// はじめの準備（3D）
void setup() {
  size(800, 600, P3D);
}

// 毎フレームすること
void draw() {
}
```

```
// はじめの準備（全画面2D）
void setup() {
  fullSrceen(P2D);
}

// 毎フレームすること
void draw() {
}
```

基本処理

setup()	最初に一度だけ呼ばれる初期化イベント関数。	void setup() { }
draw()	毎フレーム呼ばれる描画イベント関数。	void draw() { }
size()	ウィンドウの大きさと描くモードを設定する。 renderer（描くモード）：P2D, P3D, PDF	size(w, h) size(w, h, renderer)
fullScreen()	全画面表示にして描くモードを設定する。 renderer（描くモード）：P2D, P3D, PDF	fullscreen(w, h) fullscreen(w, h, renderer)

環境情報

width	画面（アプリケーションウィンドウ内）の横幅。	width
height	画面（アプリケーションウィンドウ内）の高さ（縦幅）。	height
frameCount	現在までの累計フレーム数（draw()した回数）。	frameCount
frameRate()	1秒間に何回draw()するかを設定。	frameRate(fps)
frameRate	現在1秒間に何回draw()できているかのフレームレート。	frameRate

コンソール

print()	コンソールにdataの値、文字列で出力する。	print(data)
println()	コンソールにdataの値、文字列を改行付きで出力する。	println(data)

変数型

▲

基本型（プリミティブ型）

boolean	論理値（falseまたはtrue） Boolean	boolean isOK;
byte	1バイトの整数（-128～127） Byte	byte value;
char	2バイトの整数、文字（¥u0000～¥uFFFF） Character	char c = 'A';
int	4バイトの整数（-2147483648～2147483647） Integer	int value;
float	4バイトの浮動小数（±3.40282347E+38） Float	float value;
color	4バイトの色（#000000～#FFFFFF）	color col;
long	8バイトの整数 Long	long value;
double	8バイトの浮動小数 Double	double value;

書き方

```
// 変数
int a = 10;

// 定数
final int B = 10;se
```

```
// 関数
void functionA() {
}

// 戻り値、引数付き
int functionB(int x) {
 return x;
}
```

```
// クラス
class SubClass extends SuperClass {
 int a;
 SubClass(int a) {
  super(a);
  this.a = a;
 }
 void test() { }
}
```

基本文法

//	1行コメント。	// コメント
/* */	複数行コメント。	/* コメント */
;（セミコロン）	文の終わり。	;
()（括弧）	計算の優先順位変更。	(4 + 3) * 2
,（カンマ）	関数の引数や配列の要素などを区切る。	func(10, 20)
=	変数に中身を入れる（代入）。	value = 0
final	変数を定数（値を変更しない数）として宣言する。	final int CONST_VALUE = 10;

関数

関数	関数を定義するとき、左に戻り値の型、カッコ内に必要なだけ引数の型と引数の名前を書く。	int functionName(int value) {
		  return value;
		}

void	戻り値を持たないことを示す。	void
return	関数を終了し、戻り値がある場合は戻り値を返す。	return value;

クラス

class	クラス。	class MyClass { }
extends	クラスの継承（派生）。	class Sub extends MainClass { }
interface	インターフェイスクラス（実装を含まないクラス）。	interface InterfaceClass { }
implements	インターフェイスクラスを実装する。	class Sub implements InterfaceClass { }
.（ドット）	メンバアクセス。	array.length
super	継承元（派生元）クラスのメンバーにアクセスする。	super.member()
this	自クラスのメンバーにアクセスする。	this.member()
new	クラスや配列のインスタンス（実体）を作る。	MyClass myClass = new MyClass();

ライブラリ

import	ライブラリを読み込む。	import processing.pdf.*;

定数

true	偽の論理値。	true
false	真の論理値。	false
null	未初期化、無効なデータを示す値。	null
PI	円周率（3.1415926…）。	PI
TWO_PI	円周率の2倍（6.2831853…）。	TWO_PI
HALF_PI	円周率の半分（1.5707963…）。	HALF_PI

流れ

```
if(A==B) {
  // AがBのときの処理
} else {
  // AがBでないときの処理
}
```

```
X = A==B ? true : false;
```

```
for(int i=0; i<10; i++) {
  if(A==B) continue;
}
```

```
while(true) {
  if(A==B) break;
}
```

```
switch(A) {
  case 0:
    // Aが0のときの処理
    break;
  default:
    // その他のときの処理
    break;
}
```

条件分岐

if	特定の条件に当てはまっているかどうかを判定する。	if(value < 50) { }
else	ifの条件に当てはまらなかった場合。	else { }
break	最も内側のfor、white、switchから外側に抜け出す。	break;
continue	最も内側のfor、whiteのループで、あと処理を省いて続行する。	continue;
switch	数値の値ごとの処理をcaseで書いて分岐する。	switch { }
case	switchの特定のケースを辿るラベル。	case 10:
default	switchで他のcaseに当てはまらなかったときのラベル。	default:
? :	三項演算子（ifの短縮形）。	int result = (value < 50) ? 0 : 255;

繰り返し

for	一定回数の繰り返し処理を行う。	for(int i=0; i<10; i++) { }
while	一定条件の繰り返し処理を行う。	while(value < 50) { }

計算

比較

!=	等しくない　≠	A != B
<	より小さい	A < B
<=	より小さいか等しい（以下）	A <= B
==	等しい	A == B
>	より大きい	A > B
>=	より大きいか等しい（以上）	A >= B
!	でなければ（論理否定）、NOT。	!A
&&	かつ（論理積）、AND。	A && B
||	または（論理和）、OR。	A || B

計算

+	aにbを足した値を返す。文字列のときは二つの文字列をつなげた新しい文字列を返す。	a + b string + "name"
++	1足した値を代入する。	value++ ++value
+=	aにbを足した値をaに代入する。 文字列の場合はつなげた文字列を代入する。	a += b string += "name"
-	aからbを引いた値を返す。	a - b -value
--	1引いた値を代入する。	value-- --value
-=	aからbを引いた値をaに代入する。	a -= b
*	aにbを掛けた値を返す。	a * b
*=	aにbを掛けた値をaに代入する。	a *= b
/	aをbで割った値を返す。	a / b
/=	aをbで割った値をaに代入する。	a /= b
%	aをbで割った余りを返す。	a % b

文字列

String	文字列クラス。	String string;
charAt()	指定番目の文字を返す。	.charAt(index)
indexOf()	指定の文字または文字列が現れた最初の位置を返す。	.indexOf(ch) .indexOf(ch, fromIndex) .indexOf(string) .indexOf(string, fromIdex)
equals()	文字列を比較して一致すればtrueを返す。	.equals(string)
length()	文字列の長さ(文字数)を返す。	.length()
substring()	beginIndex番目からendIndex未満の部分文字列を返す。	.substring(beginIndex) .substring(beginIndex, endIndex)
toLowerCase()	小文字に変換した文字列を返す。	.toLowerCase()
toUpperCase()	大文字に変換した文字列を返す。	.toUpperCase()

文字列関数

loadStrings()	テキストファイルを読み込み行ごとに分けたString配列を返す。 String lines[] = loadStrings("list.txt")	loadStrings(filename) loadStrings(reader)
saveStrings()	文字列をテキストファイルとして保存する。	saveStrings(filename, string)
match()	正規表現にマッチした指定グループ(括弧でマッチしたもの)を文字列の配列として返す。	match(string, regexp)
matchAll()	正規表現にマッチした指定グループ(括弧でマッチしたもの)すべてを順に格納した文字列の2次元配列を返す。	matchAll(string, regexp)
nf()	数値を指定桁数の文字列に変換する。leftで小数点より上の桁数、rightで小数点より下の桁数を指定可。	nf(num, digits) nf(num, left, right)
nfc()	数値を文字列に変換する。その際、1000の桁ごとにコンマ , 記号を付ける。	nfc(num) nfc(num, right)
split()	文字列を指定の区切り文字で分割した文字列の配列を返す。	split(string, delim)
splitTokens()	文字列を指定の区切り文字で分割した文字列の配列を返す。 delimには複数の文字を指定可能。 何も指定しなかったときは空白で分割する。	splitTokens(string) splitTokens(string, delim)
join()	文字列の配列の各要素に、seoaratorの文字または文字列を間に入れて結合した単一の文字列を返す。	join(stringArray, separator)
trim()	文字列の最初と最後から空白を除いた新しい文字列を返す。	trim(string) trim(stringArray)

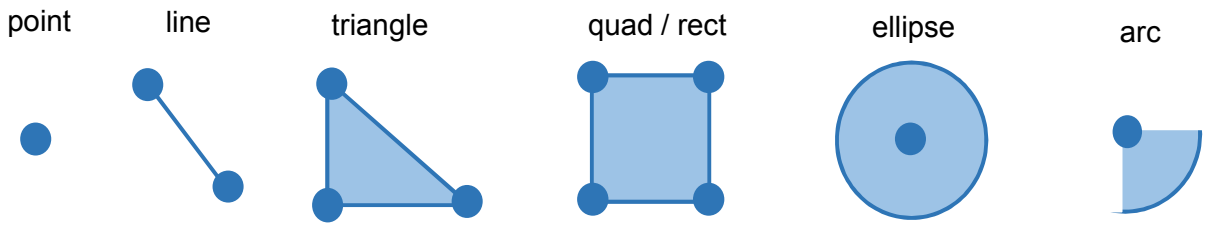
テキスト

text()	テキストを指定座標に描く。	text(str, x, y) text(str, x, y, z)
textMode()	テキストの描くモードを設定。デフォルトはMODEL。 レンダラーがP3DまたはPDFのときは、SHPAEを使用できる	textMode(mode)
textAlign()	テキストをそろえる基準を設定。 alignX(水平方向) : LEFT, CENTER, RIGHT alignY(垂直方向) : TOP, BOTTOM, CENTER, BASELINE	textAlign(alignX) textAlign(alignX, alignY)
textLeading()	テキストの行間を設定。	textLeading(leading)
textSize()	テキストサイズをピクセル単位で設定。	textSize(size)
textWidth()	テキストの幅を返す。	textWidth(c) textWidth(str)

フォント

PFont	Processing用のフォントクラス。	PFont font
createFont()	フォントを読み込んでPFontクラスに変換して返す。 インストールされているフォント(PFont.list()で確認できる)、または".ttf" と ".otf" ファイルを指定可能。	createFont("fontName", size) createFont("fontName", size, smooth) createFont("fontName", size, smooth,
loadFont()	フォントデータを読み込んだPFontクラスを返す。 (フォントデータはツールメニューの"Create Font..." で作成)	loadFont("filename.vlw")
textFont()	テキスト描くに使用するフォント(PFont)を設定。	textFont(font) textFont(font, size)

図形



2D図形

point()	点を描く。	point(x, y) point(x, y, z)
line()	直線を描く。	line(x1, y1, x2, y2) line(x1, y1, z1, x2, y2, z2)
triangle()	三角形を描く。	triangle(x1, y1, x2, y2, x3, y3)
quad()	四角形を描く。	quad(x1, y1, x2, y2, x3, y3, x4, y4)
rect()	四角形（矩形）を描く。 rに角度を指定すると、角丸で描くできる。 tl、tr、br、blで四隅の角丸の角度を指定できる。	rect(x, y, width, height) rect(x, y, width, height, r) rect(x, y, width, height, tl, tr, br, bl)
ellipse()	円を描く。	ellipse(x, y, width, height)
arc()	円弧を描く。 mode（閉じ方）：OPEN, CHORD, PIE	arc(x, y, width, height, radian_start, radian_stop) arc(x, y, width, height, radian_start,

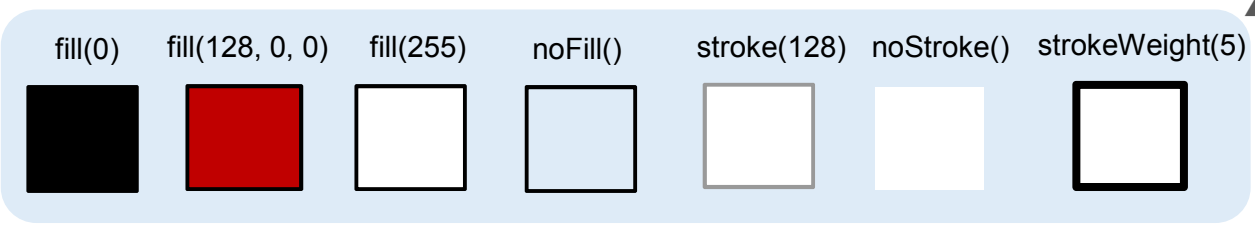
3D図形

box()	3次元の箱を描く。	box(size) box(w, h, d)
sphere()	3次元の球を描く。	sphere(radius_angle)
sphereDetail()	sphere()で描く3次元球面の細かさを指定する。 指定の角度毎に球を分割する。	sphereDetail(res) sphereDetail(ures, vres)

図形属性

smooth()	線のエッジをスムーズにする。 level（アンチエイリアス処理の量）：2, 4, 8	smooth() smooth(level)
noSmooth()	線のエッジをスムーズにしないようにする。	noSmooth()
rectMode()	rect()で矩形を描く際の座標基準を設定。 mode（座標基準）： CORNER（左、上、幅、高さ）, CORNERS（左、上、右、下）, CENTER（中央X、中央Y、幅、高さ）	rectMode(mode)
ellipseMode()	ellipse()で円を描く際の座標基準を設定。 mode（座標基準）： CORNER（左、上、幅、高さ）, CORNERS（左、上、右、下）, CENTER（中央X、中央Y、直径幅、直径高さ）, RADIUS（中央X、中央Y、半径幅、半径高さ）	ellipseMode(mode)
strokeWeight()	線の太さ（幅）を設定する。	strokeWeight(weight)
strokeCap()	線の末端の形を設定する。 cap（末端の形状）：SQUARE, PROJECT, ROUND	strokeCap(cap)
strokeJoin()	図形の角の形を設定する。 joint（角のタイプ）：MITER, BEVEL, ROUND	strokeJoin(join)

色



基本設定

background()	画面全体を背景色で塗りつぶす。 createGraphics()で作ったオフスクリーンのみ透明色を使用可。	background(rgb) background(gray) background(r, g, b) background(image)
colorMode()	色の解釈のモードを設定。 background()、color()、fill()、stroke()などの引数に影響。 mode(色の解釈):RGB(赤、緑、青)、HSV(色相、彩度、明度) ※HSVにした場合、各関数の例のr、g、b部分がh、s、vになる。	colorMode(mode) colorMode(mode, max) colorMode(mode, max1, max2, max3) colorMode(mode, max1, max2, max3, maxA)
fill()	図形の塗りつぶし色を設定。	fill(rgb) fill(rgb, alpha) fill(gray) fill(gray, alpha) fill(r, g, b) fill(r, g, b, alpha)
noFill()	図形の塗りつぶしを無効にする。	noFill()
stroke()	図形の線の色を設定。	stroke(rgb) stroke(rgb, alpha) stroke(gray) stroke(gray, alpha) stroke(r, g, b) stroke(r, g, b, alpha)
noStroke()	図形の線を無効にする(線を描かなくする)。	noStroke()
pushStyle()	色や太さなどのスタイルをスタックに1つ保存。	pushStyle()
popStyle()	色や太さなどのスタイルをスタックから1つ取り出す。	popStyle()
blendMode()	ピクセルを描く際の合成(ブレンド)モードを設定する。 mode(合成モード): BLEND(通常), ADD(加算), SUBTRACT(減算) DARKEST(比較/暗), LIGHTEST(比較/明), DIFFERENCE(差の絶対値), EXCLUSION(除外), MULTIPLY(積算), SCREEN(スクリーン), REPLACE(置換)	blendMode(mode)

色

color()	color型変数で色を作成する。初期はRGB順の指定。 colorMode()の指定によって引数がRGBかHSBかに変わる。 color(255)はcolor(255, 255, 255)と同じ意味。16進指定もでき、 color(0, 102, 153)とcolor(#006699)と#006699は同じ意味に。	color(gray) color(gray, alpha) color(r, g, b) color(r, g, b, alpha)
red()	赤色の成分を返す。	red(rgb)
green()	緑色の成分を返す。	green(rgb)
blue()	青色の成分を返す。	blue(rgb)
alpha()	アルファ値(透明度)を返す。0が透明、255が不透明。	alpha(rgb)
brightness()	輝度(明るさ)を返す。	brightness(rgb)
hue()	色相を返す。	hue(rgb)
saturation()	彩度を返す。	saturation(rgb)
lerpColor()	c1からc2までをamt(0～1)の補間量で線形補間した色を返す。	lerpColor(c1, c2, amt)

画像

PImage	画像データクラス。	Pimage image
width	画像の横幅。	.width
height	画像の高さ(縦幅)。	.height
resize()	pixels[]配列のピクセルデータを、現在の画面に反映させる。	.resize(w, h)
get()	特定のピクセルに色を設定する。pixels[]よりは低速。 PImageの場合は左上座標を指定する(imageMode()の影響は受けない)。	.get(x, y) .get(x, y, w, h) .get()
set()	特定のピクセルに色を設定する。pixels[]よりは低速。 PImageの場合は左上座標を指定する(imageMode()の影響は受けない)。	.set(x, y, c) .set(x, y, img)
save()	画像をファイルに保存する。 拡張子でファイル形式を指定(jpg、png、tga、tif)。	.save(filename)

画像関数

loadImage()	画像ファイルを読み込み、PImageクラスを返す。 形式は.gif, .jpg, .tga, .pngに対応。	loadImage(filename) loadImage(filename, extension)
createImage()	画像(PImage)を生成して返す。 format(画像バッファタイプ): RGB, ARGB, ALPHA	createImage(w, h, format)
save()	画面のスクリーンショットをファイルに保存する。 拡張子でファイル形式を指定(jpg、png、tga、tif)。	save(filename)
saveFrame()	画面のスクリーンショットを連番付きファイルに保存する。 saveFrame("scree####.png")のように#で連番位置を指定可。	saveFrame() saveFrame(filename)
image()	画像(PImage)を指定座標、指定サイズで描く。	image(image, x, y) image(image, x, y, width, height)
imageMode()	image()で画像を描く際の座標基準を設定。 mode(座標基準): CORNER(左、上、幅、高さ), CORNERS(左、上、右、下), CENTER(中央X、中央Y、幅、高さ)	imageMode(mode)
tint()	image()などの画像描くの塗りつぶし補正色を設定する。 tint(255)で画像そのものの色、 tint(255, 128)で半透明描くになる。	tint(rgb) tint(rgb, alpha) tint(gray) tint(gray, alpha) tint(r, g, b) tint(r, g, b, alpha)
noTint()	tint()による塗りつぶし補正色を無効にする	noTint()

ピクセル

copy()	画面または画像の一部を画面の別の場所にコピーする。	copy(sx, sy, sw, sh, dx, dy, dw, dh) copy(srcImage, sx, sy, sw, sh, dx, dy, dw, dh)
get()	特定のピクセルの色を取得する。pixels[]よりは低速。 get()のみの場合は現在の画面のPImageを、 get(x, y, w, h)の場合は切り出した部分のPImageを返す。	get(x, y) get(x, y, w, h) get()
set()	特定のピクセルに色を設定。pixels[]よりは低速。PImageの場合は左上座標を指定する(imageMode()の影響なし)。	set(x, y, col) set(x, y, image)

特殊

createGraphics()	PGraphicsクラスを生成して返す。size()関数と同様の初期化。	createGraphics(w, h) createGraphics(w, h, renderer) createGraphics(w, h, renderer, path)
PGraphics	Processingのline()、fill()などコアAPIを使える画面クラス。 オフスクリーン(見えないサブ画面)に描くときに使用。 PGraphicsは透明度を持てる。	PGraphics pg
beginDraw()	オフスクリーンへの描く開始。	beginDraw()
endDraw()	オフスクリーンへの描く終了。オフスクリーンpgを実際の画面に描くには、image(pg, 0, 0)とする。	endDraw()

操作

マウス/タッチ

mouseX	現在のマウスのX座標。	mosueX
mouseY	現在のマウスのY座標。	mouseY
pmouseX	1フレーム前のマウスのX座標。	pmouseX
pmouseY	1フレーム前のマウスのY座標。	pmouseY
mousePressed	マウスボタンが押されている状態ならtrueになる変数。	mousePressed
mouseButton	マウスのボタンが押されたとき、どのボタンかを示す変数。 LEFT, RIGHT, CENTERのどれか。	mouseButton
mousePressed()	マウスボタンを押したときに呼ばれるイベント関数。	mousePressed() mousePressed(event)
mouseReleased()	マウスボタンを離したときに呼ばれるイベント関数。	mouseReleased() mouseReleased(event)
mouseClicked()	マウスでクリック時に呼ばれるイベント関数。	mouseClicked() mouseClicked(event)
mouseMoved()	マウスのボタンを押さずに動かしたときのイベント関数。	mouseMoved() mouseMoved(event)
mouseDragged()	マウスをドラッグ時（ボタンを押したまま動かしたとき）に呼ばれるイベント関数。	mouseDragged() mouseDragged(event)

キーボード

key	直前に押されていたキーコードを記憶した変数。 Aキーであれば、'a'や'A'と比較して判定できる。	key
keyCode	keyがCODEDのときに、次のような特殊なキーを判別する。 （UP, DOWN, LEFT, RIGHT, ALT, CONTROL, SHIFT）	keyCode
keyPressed	何かキーが押されている状態ならtrueになる変数。	keyPressed
keyPressed()	キーを押したときに呼ばれるイベント関数。	keyPressed() keyPressed(event)
keyReleased()	キーを離したときに呼ばれるイベント関数。	keyReleased() keyReleased(event)
keyTyped()	キーを1文字すごとに呼ばれるイベント関数。	keyTyped() keyTyped(event)

座標変換

translate()	平行移動する。	translate(x, y) translate(x, y, z)
rotate()	指定の角度（ラジアン単位）だけ回転する（rotateZ()と同じ）。 x, y, z付きだとそれを軸として回転する。	rotate(radian_angle) rotate(radian_angle, x, y, z)
rotateX()	X軸を周りに指定の角度（ラジアン単位）だけ回転する。	rotateX(radian_angle)
rotateY()	Y軸を周りに指定の角度（ラジアン単位）だけ回転する。	rotateY(radian_angle)
rotateZ()	Z軸を周りに指定の角度（ラジアン単位）だけ回転する。	rotateZ(radian_angle)
scale()	拡大縮小する。	scale(s) scale(x, y) scale(x, y, z)
pushMatrix()	現在の行列（移動、回転、拡大縮小などの情報）を1つ積み上げて覚えておく。	pushMatrix()
popMatrix()	pushMatrix()で一番上に積み上げた行列（移動、回転、拡大縮小などの情報）を1つ取り除いて、現在の行列に設定する。	popMatrix()

座標系変換

modelX()	モデル空間座標を原点からの座標に変換、そのX座標を返す。	modelX(x, y, z)
modelY()	モデル空間座標を原点からの座標に変換、そのY座標を返す。	modelY(x, y, z)
modelZ()	モデル空間座標を原点からの座標に変換、そのZ座標を返す。	modelZ(x, y, z)
screenX()	3D座標を2Dスクリーン上の座標に変換、そのX座標を返す。	screenX(x, y, z)
screenY()	3D座標を2Dスクリーン上の座標に変換、そのY座標を返す。	screenY(x, y, z)
screenZ()	3D座標を2Dスクリーン上の座標に変換、そのZ座標を返す。	screenZ(x, y, z)

日時

day()	現在の日(1～31)を返す。	day()
hour()	現在の時(0～23)を返す。	hour()
millis()	プログラムを起動してからの経過ミリ秒を返す。1000で1秒。	millis()
minute()	現在の分(0～59)を返す。	minute()
month()	現在の月(1～12)を返す。	month()
second()	現在の秒(0～59)を返す。	second()
year()	現在の年(2013、2014、…etc.)を返す。	year()

データ

コレクションクラス(コンポジット型)

Array	配列。	int[] array = { 10, 20, 30 };
[] (配列アクセス)	配列アクセス。	array[index]
length	配列の大きさ(要素数)。	array.length
ArrayList	可変長配列クラス。	ArrayList arrayList;
add()	要素を追加する。	.add(element)
get()	index番目の要素を返す。	.get(index)
indexOf()	指定要素が追加済みならそのindexを返す。無ければ-1を返す。	.indexOf(element)
clear()	要素を全削除する。	.clear()
remove()	index番目の要素を削除する。	.remove(index)
size()	追加済みの要素数を返す。	.size()
HashMap	連想配列クラス。	HashMap hashMap;
put()	指定のkeyで、指定のvalueを追加する。	.put(key, value)
get()	指定のkeyの要素を返す。無ければnullを返す。	.get(key)
containsKey()	指定のkeyの要素が追加済みならtrueを返す。	.containsKey(key)
containsValue()	指定のvalueの要素が追加済みならtrueを返す。	.containsValue(value)
clear()	要素を全削除する。	.clear()
remove()	指定のkeyの要素を削除する。	.remove(key)
size()	追加済みの要素数を返す。	.size()
Table	テーブル(表)クラス。	Table table;
addColumn()	列を追加する。	.addColumn() .addColumn(title)
addRow()	行を追加する。	.addRow() .addRow(stringArray)
getRowCount()	行数を返す。	.getRowCount()
getString()	row行、column列の値を文字列で返す。	.getString(row, column)
setString()	row行、column列に文字列を設定する。	.setString(row, column, string)
getInt()	row行、column列の値をint型で返す。	.getInt(row, column)
setInt()	row行、column列にint型の値を設定する。	.setInt(row, column, intValue)
getFloat()	row行、column列の値をfloat型で返す。	.getFloat(row, column)
setFloat()	row行、column列にfloat型の値を設定する。	.setFloat(row, column, floatValue)
createTable()	Tableクラスを作成して返す。	createTable()
loadTable()	CSVもしくはTSVファイルを読み込んでTableクラスを返す。 Optionsに"header"を指定すると最初の行をヘッダーと解釈。 "csv"でカンマ区切り、"tsv"でタブ区切りのファイル。 loadTable("data.csv", "header, tsv")	loadTable(filename) loadTable(filename, options)
saveTable()	Tableクラスのデータを指定の名前のファイルとして保存する。	saveTable(table, filename) saveTable(table, filename, options)

ライトとカメラ

ライト

lights()	ライトをデフォルト値に設定する。	lights()
noLights()	ライトを削除して無効にする。	noLights()
ambientLight()	環境光を追加する。	ambientLight(r, g, b) ambientLight(r, g, b, x, y, z)
directionalLight()	平行光源を追加する。 nx、ny、nz：光の向きを示す長さ1のベクトル	directionalLight(r, g, b, nx, ny, nz)

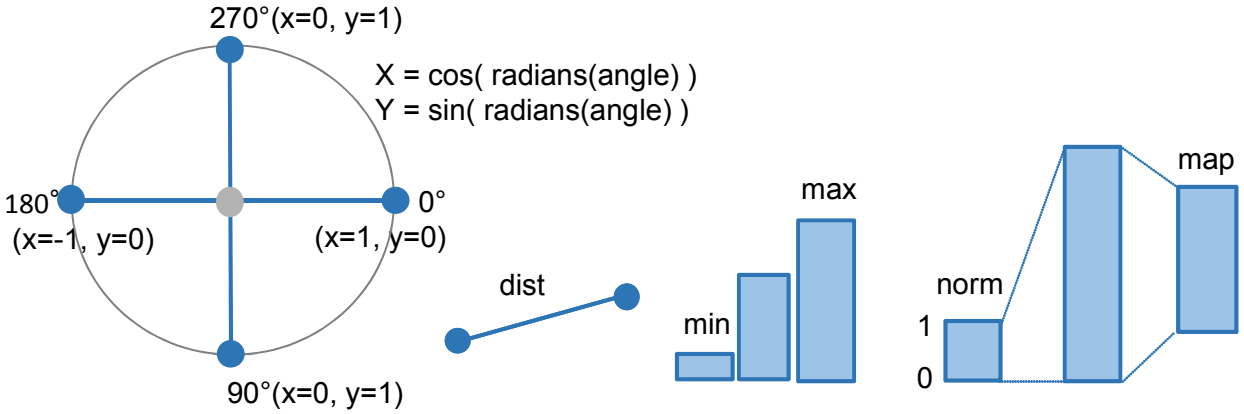
カメラ

camera()	カメラを位置と向きを設定する。 eyeX, eyeY, eyeZ：カメラを置く位置 targetX, targetY, targetZ：カメラが目を向けるターゲット位置 upX, upY, upZ：カメラの上方向を示す長さ1のベクトル	camera() camera(eyeX, eyeY, eyeZ, targetX, targetY, targetZ, upX, upY, upZ)
perspective()	透視射影で遠近感あるカメラモードを設定する。 perspective(画角, アスペクト比, 見える最小距離, 最大距離)	perspective() perspective(radian_fovy, aspect, zNear, zFar)
ortho()	正射影で遠近感のないカメラモードを設定する。 ortho(左, 右, 下, 上, 近, 遠)	ortho() ortho(left, right, bottom, top) ortho(left, right, bottom, top, near, far)
frustum()	透視射影で遠近感あるカメラモードを設定する。 frustum(左, 右, 下, 上, 近, 遠)	frustum(left, right, bottom, top, near, far)

ベクトル →

PVector	Processing用のベクトルクラス。	Pvector vector;
x, y, z	ベクトルのX、Y、Z要素。	.x .y .z
set()	ベクトルのX、Y、Z要素に値を代入する。	.set(x, y, z) .set(vector) .set(floatArray)
get()	同じ値を持つ新しいPVectorクラスを返す。	.get() .get(target)
fromAngle()	指定の角度（ラジアン単位）の2D単位ベクトルを返す。	.fromAngle(radian_angle) .fromAngle(radian_angle, target)
mag()	ベクトルの長さを返す。	.mag()
add()	ベクトルに指定の値を足す。	.add(vector) .add(x, y, z) .add(v1, v2) .add(v1, v2, target)
sub()	ベクトルから指定の値を引く。	.sub(vector) .sub(x, y, z) .sub(v1, v2) .sub(v1, v2, target)
mult()	ベクトルをn倍する。	.mult(n) .mult(vector, n) .mult(vector, n, target)
div()	ベクトルをnで割る。	.div(n) .div(vector, n) .div(vector, n, target)
dist()	2点間の距離を返す。	.dist(vector) .dist(v1, v2)
dot()	ベクトルの内積を返す。	.dot(vector) .dot(x, y, z) .dot(v1, v2)
cross()	ベクトルの外積を返す。	.cross(v) .cross(v, target) .cross(v1, v2, target)
normalize()	単位ベクトル（向きはそのままに長さが1のベクトル）にする。	.normalize() .normalize(target)
lerp()	2つのベクトルをamt（0～1）で線形補間した値を返す。	.lerp(vector, amt) .lerp(v1, v2, amt) .lerp(x, y, z, amt)

数学



abs()	絶対値（正の値）を返す。	abs(n)
round()	小数点以下を四捨五入した整数を返す。	round(value)
ceil()	小数点以下を切り上げした値を返す。	ceil(n)
floor()	小数点以下を切り捨てした値を返す。	floor(n)
min()	最少の値を返す。	min(a, b) min(a, b, c) min(array)
max()	最大の値を返す。	max(a, b) max(a, b, c) max(array)
constrain()	valueをlow～highの範囲内に規制した値を返す。	constrain(value, low, high)
map()	valueがstart1～stop1のときstart2～stop2になる値を返す。	map(value, start1, stop1, start2, stop2)
norm()	valueがstart～stopのとき0～1になる値に正規化して返す。	norm(value, start, stop)
lerp()	startからstopまでをamt（0～1）で線形補間した値を返す。	lerp(start, stop, amt)
mag()	ベクトル（原点からx, yまで）の長さを返す。	mag(x, y) mag(x, y, z)
dist()	2点間の距離を返す。	dist(x1, y1, x2, y2) dist(x1, y1, z1, x2, y2, z2)
exp()	自然対数の底eをn乗した値を返す。	exp(n)
log()	自然対数を返す。	log(n)
pow()	nのe乗を返す。	pow(n, e)
sq()	二乗を返す。	sq(value)
sqrt()	平方根（ルート）を返す。	sqrt(value)

三角関数

radians()	角度をラジアン値に変換。 360 → 2 * PI	radians(degree_angle)
degrees()	ラジアン値を角度に変換。 2 * PI → 360	degrees(radian_angle)
sin()	サイン値を返す（-1から1）。	sin(radian_angle)
cos()	コサイン値を返す（-1から1）。	cos(radian_angle)
tan()	タンジェント値を返す。	tan(radian_angle)
asin()	sin()の逆、アークサイン値を返す（-PI/2からPI/2）。	asin(value)
acos()	cos()の逆、アークコサイン値を返す（0からPI）。	acos(value)
atan()	tan()の逆、アークタンジェント値を返す（-PI/2からPI/2）。	atan(value)
atan2()	座標から角度（ラジアン単位）、アークタンジェント値を返す（PIから-PI）。	atan2(y, x)

ランダム（乱数）

random()	一定範囲の乱数を返す。	random(high) random(low, high)
randomSeed()	random()用の乱数の種を設定し、乱数系列を初期化。	randomSeed(seed)
noise()	指定された座標のPerlinノイズ値を返す。	noise(x) noise(x, y) noise(x, y, z)
noiseDetail()	noise()用のオクターブと減衰量を設定。	noiseDetail(octaves) noiseDetail(octaves, falloff)
noiseSeed()	noise()用の乱数の種を設定し、乱数系列を初期化。	noiseSeed(seed)