

Processing Quickstart

Screen

height // find out height of screen

width // find out width of screen

size(h, w); // sets the screen dimensions

background(0); // clears screen to black

Coordinates: location on the screen, written as (x,y); **x** is the distance across (from left) and **y** is the distance down (from top)

distance between points (x1,y1) and (x2,y2):
 $\text{sqrt}((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2));$

Drawing

point(x,y); // draws a point at x,y

line(x1,y1,x2,y2); // draws a line from (x1,y1) to (x2,y2)

triangle(x1,y1,x2,y2,x3,y3); // draws a triangle with the three points as corners

rect(x1,y1,height,width); // draws a rectangle centred at (x,y) with the given height & width

ellipse(x,y,height,width); // draws an ellipse at (x,y) – note, an ellipse with equal height and width is a circle

Color

RGB: red, green, blue, values 0-255; e.g., Red: (255,0,0); Green: (0,255,0); Blue: (0,0,255); Yellow: (255,255,0);

fill(r,g,b); // shapes will be filled with given color

stroke(r,g,b); // lines will be drawn with given color

strokeWeight(w); // sets line thickness

Interaction

mouseX // x coordinate of cursor

mouseY // y coordinate of cursor

mousePressed // is a mouse button currently pressed?

Variables

int: whole number type (integer)

float: decimal number type

declaring a variable: “int x;” – then “**x**” is available to store an integer quantity

PImage: Processing image type

assignment: equals sign “=”; e.g., x = 4 assigns variable x the value 4

Image

save(filename); // saves screenshot in sketch folder

myimage = load(“image.jpg”); // loads an image;
remember to declare myimage and import image.jpg

image(myimage, x, y); // draws the image at (x,y)

image(myimage, x, y, image_height, image_width); // draws the image with dimensions (image_height, image_width);

loadPixels(); // loads the screen into “pixels”

updatePixels(); // resets the screen according to “pixels”

pixels[which] // gives you access to pixel #which, counting across starting from the top left corner

Structure

setup() { ... } // these commands are done once

draw() { ... } // these commands are done over and over

Control

if (condition) statement; // *statement* is executed if *condition* is true

if (condition) statement1;
else statement2; // *statement1* is executed if *condition* is true, otherwise *statement2* is executed

while (condition) statement; // *statement* is executed over and over as long as *condition* is true

for (int i = 0; i < 1000; i++) statement; // execute *statement* 1000 times

for (initialization; condition; increment) statement; // first execute *initialization*, and execute *statement* and *increment* over and over as long as *condition* is true

Logic

Conditions are statements that are either true or false

equality: (x == y) // true if x and y are equal

inequality: (x != y) // true if x and y are different

greater than: (x > y)

less than: (x < y)

greater than or equal to: (x >= y)

less than or equal to: (x <= y)

AND: (condition1 && condition2) // true if both *condition1* and *condition1* are true, otherwise false

OR: (condition1 || condition2) // true if at least one of *condition1* or *condition2* is true, false if both are false

NOT: (!condition) // true if *condition* is false, false if *condition* is true

Math

+, -, *, /, % : addition, subtraction, multiplication, division, remainder

sqrt(x) // square root of x

sq(x) // square of x, i.e., x*x

max(x,y) // larger of x and y

min(x,y) // smaller of x and y

abs(x) // absolute value of x

random(high) // random number less than *high*

random(low, high) // random number at least *low* and less than *high*

radians(angle) // convert degrees to radians

sin(angle) // sine of *angle*

cos(angle) // cosine of *angle*

Turtle

You must have created a turtle *t* to use these.

t.forward(x); // move forward x

t.backward(x); // back up x

t.right(x); // turn right x degrees

t.left(x); // turn left x degrees

t.gotopos(x,y); // move to location (x,y)

More help

www.processing.org/reference