Common Processing Commands
CMSC 101 / IS 101Y - Fall 2013

This document summarizes the commands that you are most likely to use in your Processing assignments and final project. To learn about many more useful Processing commands, refer to the textbook and to http://processing.org/reference/ .

Program Control: *The following functions are used to control logic*

if(condition){//cmds}:	*Runs commands only when condition is true (called a conditional)*
else{//cmds}:	*Used with an if statement. Runs if the if statement doesn't run*
else if(condition){//cmds}:	*Used to have another conditional after one fails.*
for(init; condition; update){//cmds}:	*Perform "init"; test "condition" before loop; perform "cmds"; perform "update"*
for(int i = 0; i < list.length; i++){//cmds}:	*Specific example: through code until middle condition is not true. In this case, increments variable by one each iteration.*
while(condition){//cmds}:	*Runs until condition is false, controlling logic in the code block*

Logic Operators: *Useful logic operators for if, for, and while statements*

< *Less than*	<= *Less than or equal to*
> *Greater than*	>= *Greater than or equal to*
== *equals*	!= *does not equal*
&& *AND operator.*	*Returns true if all expressions are true, false otherwise*
|| *OR operator.*	*Returns true if at least one expression is true, false otherwise*
! *NOT operator.*	*Negates the statement (true becomes false and vice versa)*

Variable and Constant Declaration: *Examples of creating new variables and constants*

int x = 45;	*Declares x as an int (int x) and sets(aka instantiates) the value of x to 45 (= 45)*
float degrees;	*Declares "degrees" as a float (real-valued) variable but doesn't initialize it*
final float pi = 3.14;	*Values like pi, which never change, should be constants ("fixed")*
final int MaxImages = 100;	*Defines a constant (unchanging) value for MaxImages*
String quote = "When in the course of human events…";	*A string is a text-storage variable*
boolean example = true;	*Booleans are used to store true/false values. See Logic Operators.*
float xpos[] = new float[100];	*Defines an array of floats; creates an empty array of length 100*

Math Functions: *The following functions make geometric operations on your shapes easier*

abs (x)	*Returns the absolute value of the parameter "x"*
dist (x1, y1, x2, y2)	*Returns the Euclidean distance between points [x1,y1] and [x2,y2]*
sin (angle)	

cos (angle)	
translate (x, y)	*All images drawn after this will be moved x pixels down and y pixels right*
rotate (angle)	*All images drawn after this will be rotated clockwise by “angle” degrees*
scale (proportion)	*All images drawn after this will be rescaled by “proportion”*
int z = x + y;	*This code assumes x and y are known and declares z to store their sum*
int z = x - y;	
int z = x * y;	
int z = x / y;	*Divide and round down. If you don’t want rounding, use float variables*
int z = x % y;	*Set z to be the remainder of dividing x by y*
int z = x++;	*Add one to the value of x*
int z = x --;	*Subtract one from the value of x*
random (N);	*Generate a random number between 0 and N*
random (Low,High);	*Generate a random number between Low and High*

Display and I/O: *These are general functions to control the appearance of the display and to do input and output*

size (width, height)	*Change the size of the display window*
background (color)	*Change the background color of the display window*
print (arg arg …)	*Print a message in the console window*
println (arg arg …)	*Print a message in the console window, then a new line*
millis ()	*Current time since the program started, in milliseconds*

Standard Functions: *These are built-in functions that you can define to control the behavior of your program*

setup ()	*Executed once when the program is started*
draw ()	*Executed repeatedly and automatically to update the display, frameRate times per second*
keyPressed ()	*Called whenever any key is pressed*
mousePressed ()	*Called whenever a mouse button is pressed*
mouseMoved ()	*Called whenever the mouse is moved*
exit ()	*Ends the program and closes the display window*

Shapes: *The following are the functions provided by Processing to draw shapes quickly*

line (x1, y1, x2, y2)	
ellipse (x, y, width, height)	
rect (x, y, width, height)	
triangle (x1, y1, x2, y2, x3, y3)	
beginShape (); vertex (x,y); …; endShape ();	*Draw a polygon with a series of vertices*

Line Attributes: *The following functions are used to add attributes (like color and width) to lines*

smooth () / noSmooth ()	*Turn smoothing (prettier curves) on and off*
strokeWeight (weightVal)	*All lines after this will be weightVal pixels wide*
fill (grayscale) / fill (R, G, B)	*Define a grayscale or RGB color to fill future shapes with*

User Input Variables: *The following are variables stored by Processing about user input*

mouseX / mouseY	*Current mouse position (x and y)*
pMouseX / pMouseY	*Mouse position in the previous frame (time step) (x and y)*
mousePressed	*Boolean variable that is True if any mouse button is pressed*
mouseButton	*Which of the three mouse buttons is pressed (if any)* - LEFT/CENTER/RIGHT
keyPressed	*Boolean variable that is True if any key is pressed*
key	*A character representing the key that is currently pressed (e.g., 'c')*

Text & Fonts: *The following functions can be used to display text in your window. Once the text has been drawn, text can be treated like a shape with regards to size, direction, etc.*

```
textSize ()
textAlign ()
text (key, x, y)
loadFont (filename)
textFont (fontVar)
```

Images: *The following lines will load an image into your window. From there the image can be treated like a shape with regards to size, direction, etc.*

```
imageVar = loadImage (filename);
image (imageVar, x, y);
```

Function Definition: *The following code will demonstrate defining custom functions.*
The following function has no return type ("void") and takes no arguments. It could be used to print instructions or an error message, for example.

```
        void functionName () {
                //commands go here...
        }
```

The following function will return an integer and has two arguments. Only the change made to x will be saved, because it was returned. (Note that whatever variable is sent in as "x" won't automatically be changed in the calling function!) Other changes are "local," meaning that they won't change anything once the function ends.

```
        int functionName (int x, int y) {
                //commands go here...
                y--;
                return x++;
        }
```