SM2715 (L02) Creative Coding
week 01

by Mike Wong. School of Creative Media

Evaluation

TASKS	Weighting
Coding Assignment x 3 (individual)	50% (15%, 15%, 20%)
Final Project (Group: 2 persons)	40%
In-class Exercise & Participation	10%

Policies

Attendance & Participation:

10% of total grade

more than 3 absences may fail the course

Assignment Late Submission:

10% mark deduction per day

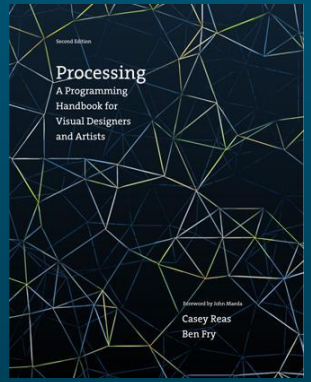
Academic Honesty / Plagiarism:

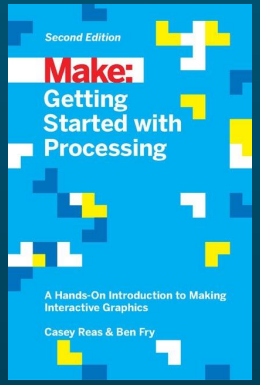
School's default penalty: **F-grade** for the course

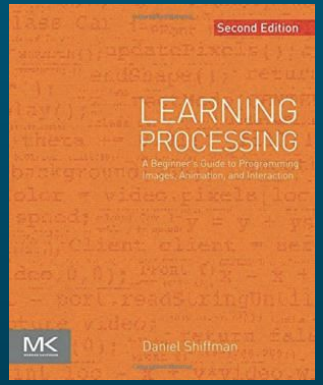
https://www6.cityu.edu.hk/ah/plagiarism.htm

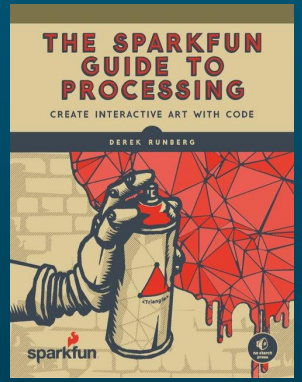
Week 01
Basics of Processing

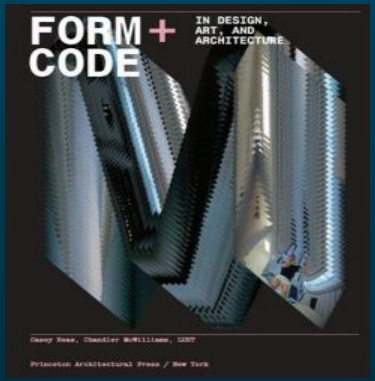
References

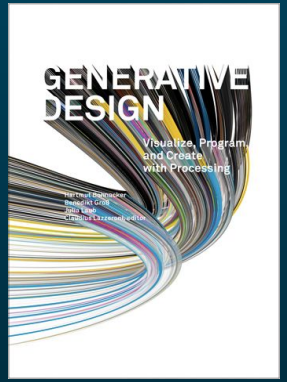

Processing
A Programming Handbook for Visual Designers and Artists
Casey Reas
Ben Fry

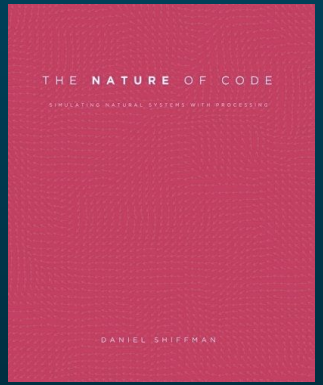

Second Edition
Make:
Getting Started with Processing
A Hands-On Introduction to Making Interactive Graphics
Casey Reas & Ben Fry

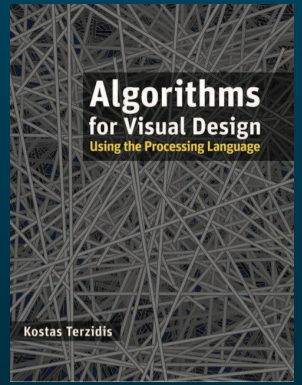

Second Edition
LEARNING PROCESSING
A Beginner's Guide to Programming Images, Animation, and Interaction
Daniel Shiffman


THE SPARKFUN GUIDE TO PROCESSING
CREATE INTERACTIVE ART WITH CODE
DEREK RUNBERG
sparkfun

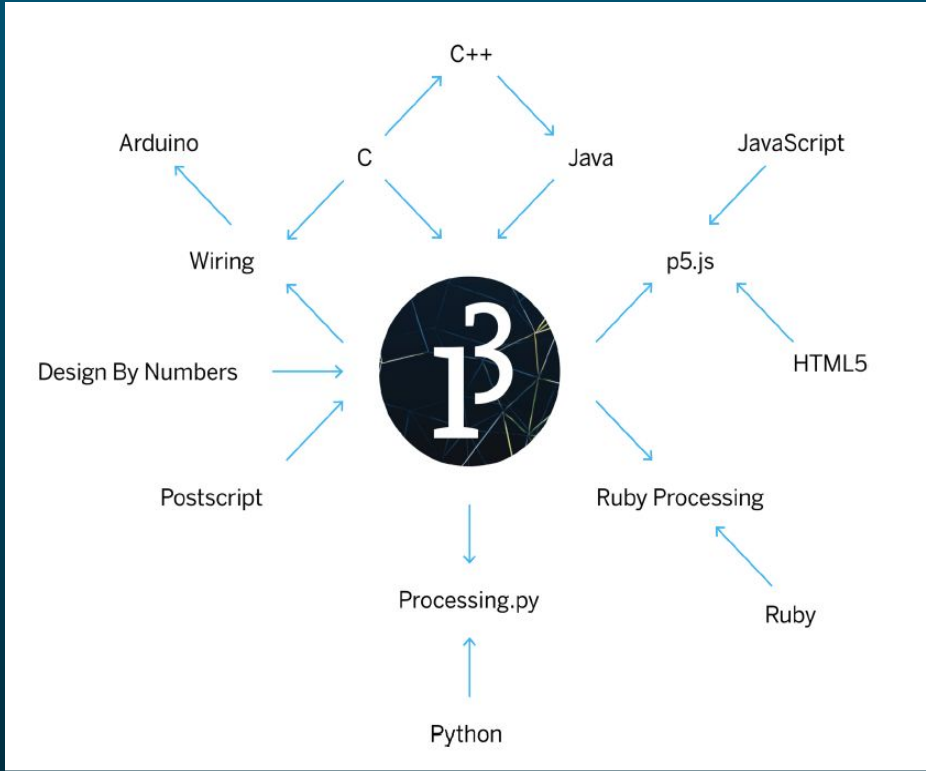

FORM + CODE
IN DESIGN, ART, AND ARCHITECTURE


GENERATIVE DESIGN
Visualize, Program, and Create with Processing


THE NATURE OF CODE
SIMULATING NATURAL SYSTEMS WITH PROCESSING
DANIEL SHIFFMAN

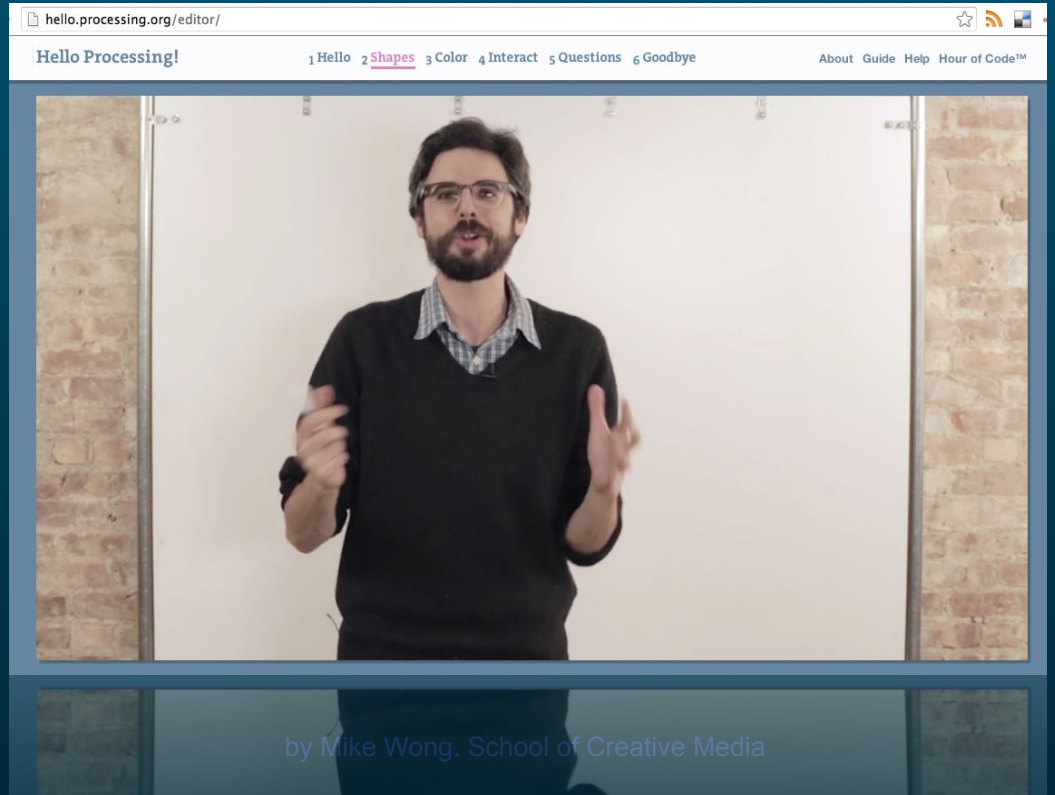

Algorithms for Visual Design
Using the Processing Language
Kostas Terzidis

Processing Family Tree



Hello Processing by Daniel Shiffman

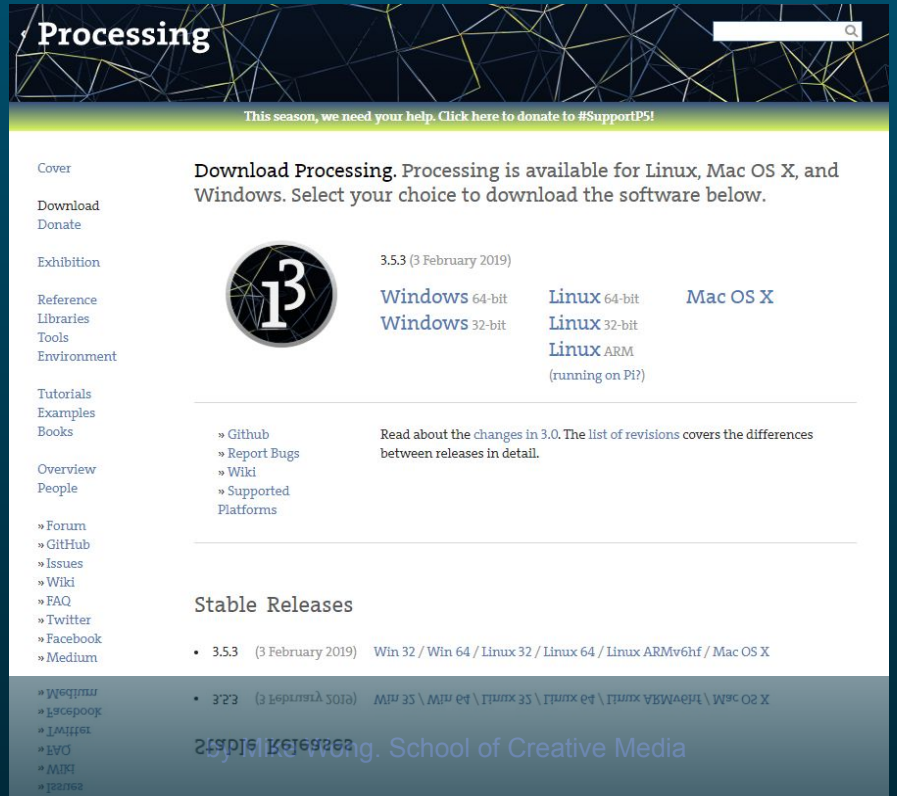
URL: https://hello.processing.org



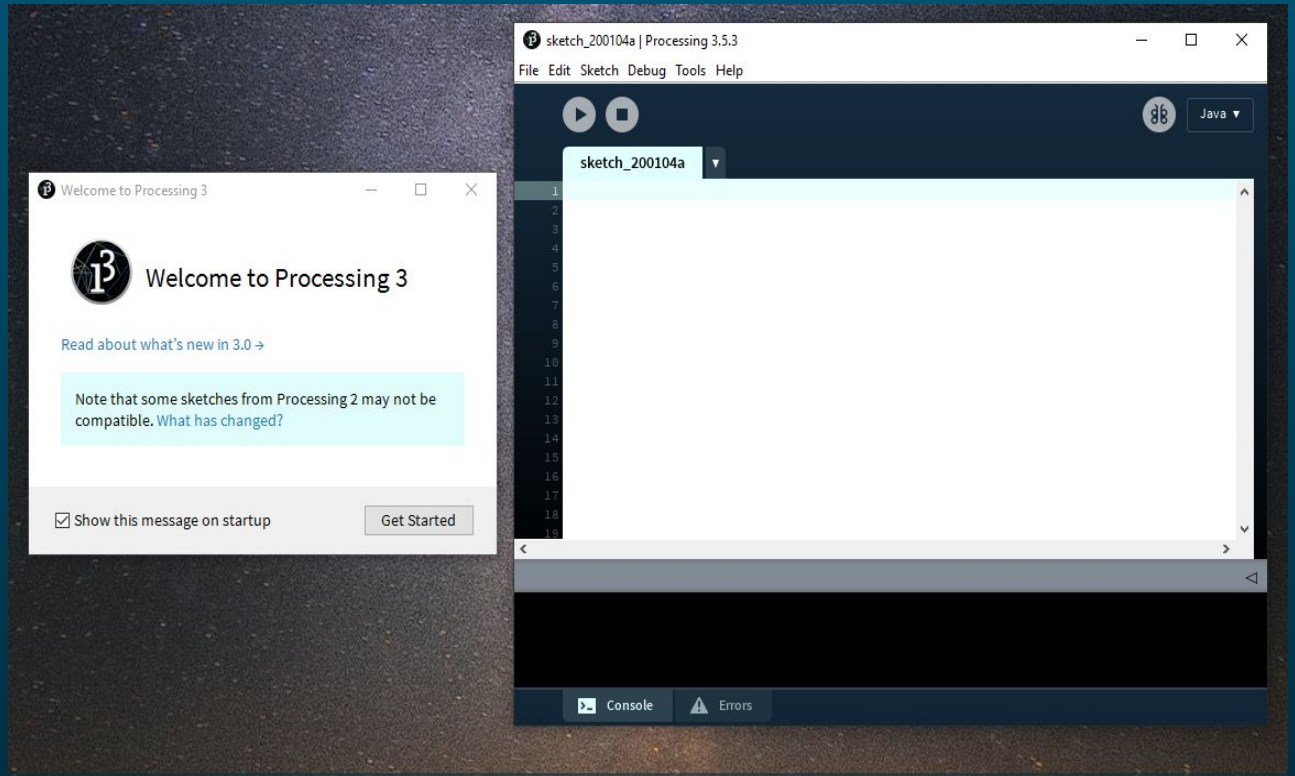
by Mike Wong. School of Creative Media

Download & Install Processing

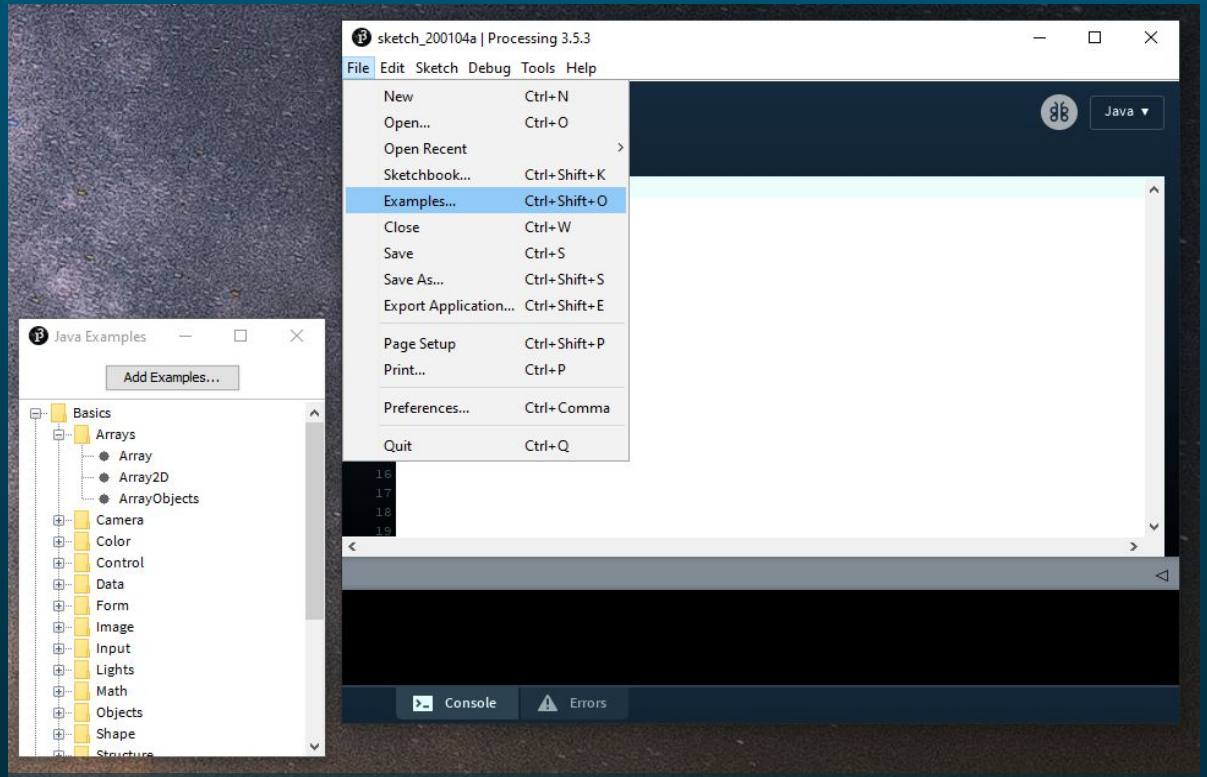
URL: https://www.processing.org/download/



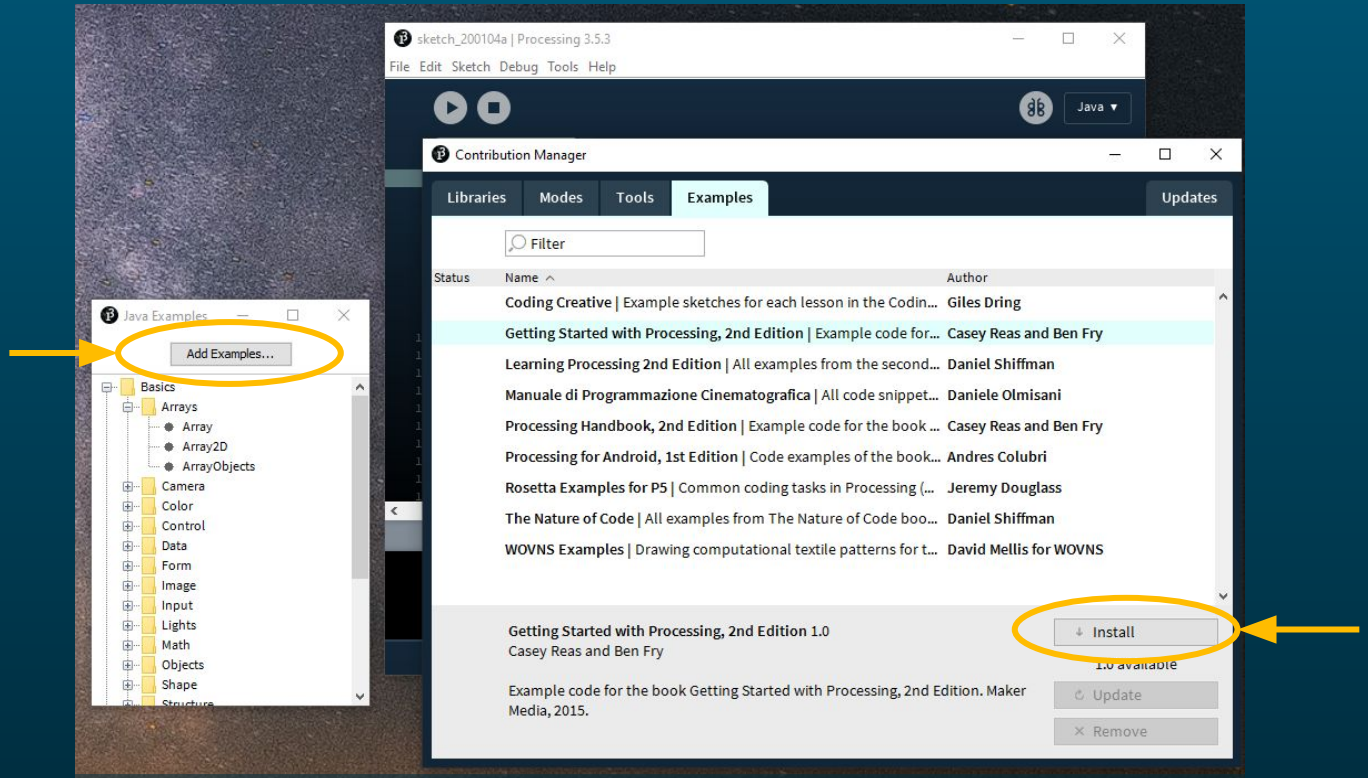
Launch Processing



To Learn by Examples



To Add More Examples



Examples on Processing.org

Processing

This season, we need your help. Click here to donate to #SupportP5!

Cover

Download
Donate

Exhibition

Reference
Libraries
Tools
Environment

Tutorials
Examples
Books

Overview
People

» Forum
» GitHub
» Issues
» Wiki
» FAQ
» Twitter

Examples. Short, prototypical programs exploring the basics of programming with Processing.

These examples are running online through p5.js using HTML Canvas for rendering. There are many more examples included with the Processing application; please look there if you don't find what you're looking for here.

Basic Examples. *Programs about form, data, images, color, typography, and more...*

Structure

Statements and Comments
Coordinates
Width and Height
Setup and Draw
No Loop
Loop
Redraw
Functions
Recursion
CreateGraphics

Form

Image

Load and Display Image
Background Image
Transparency
Alphamask
CreateImage
Pointillism
Request Image

Color

Hue
Saturation

Input

Mouse 1D
Mouse 2D
MousePress
Mouse Signals
Easing
Constrain
Storing Input
Mouse Functions
Keyboard
Keyboard Functions
Milliseconds
Clock

Reference on Processing.org

Processing

This season, we need your help! Click here to donate to #SupportP5!

Cover

Download
Donate

Exhibition

Reference
Libraries
Tools
Environment

Tutorials
Examples
Books
Overview
People

» Forum
» GitHub
» Issues
» Wiki
» Issues
» GitHub
» Forum

Reference. Processing was designed to be a flexible software sketchbook.

Structure

() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
extends
exit()
draw()
class

Shape

createShape()
loadShape()
PShape

2D Primitives
arc()
circle()
ellipse()
line()
point()
quad()
rect()
square()
triangle()

triangle()
square()
rect()

Color

Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()

Creating & Reading
alpha()
blue()
brightness()
color()
green()
green()
color()
brightness()
blue()

Getting Help (off-line)

Right click on an item to get help from the offline reference page

Letters | Processing 3.5.3

File Edit Sketch Debug Tools Help

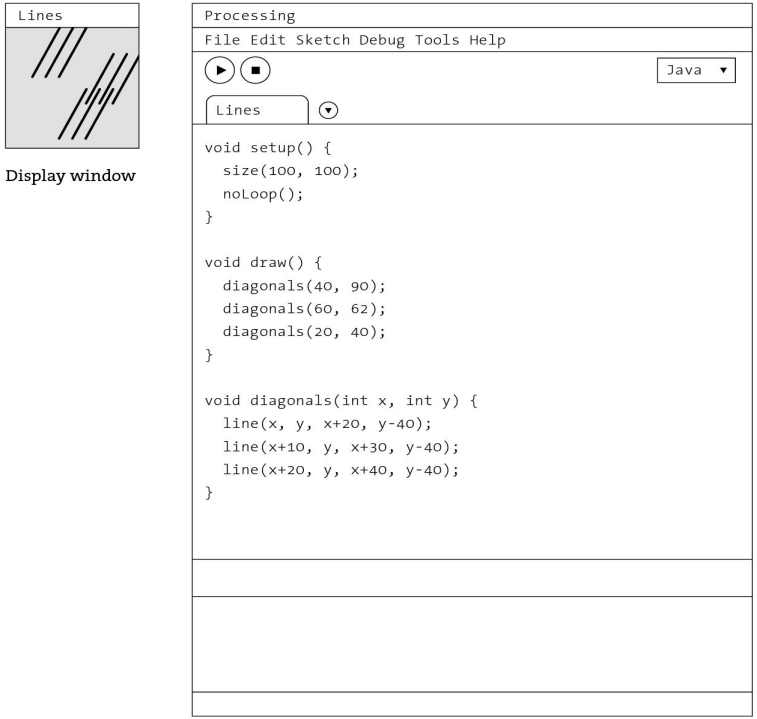
Java ▾

Letters

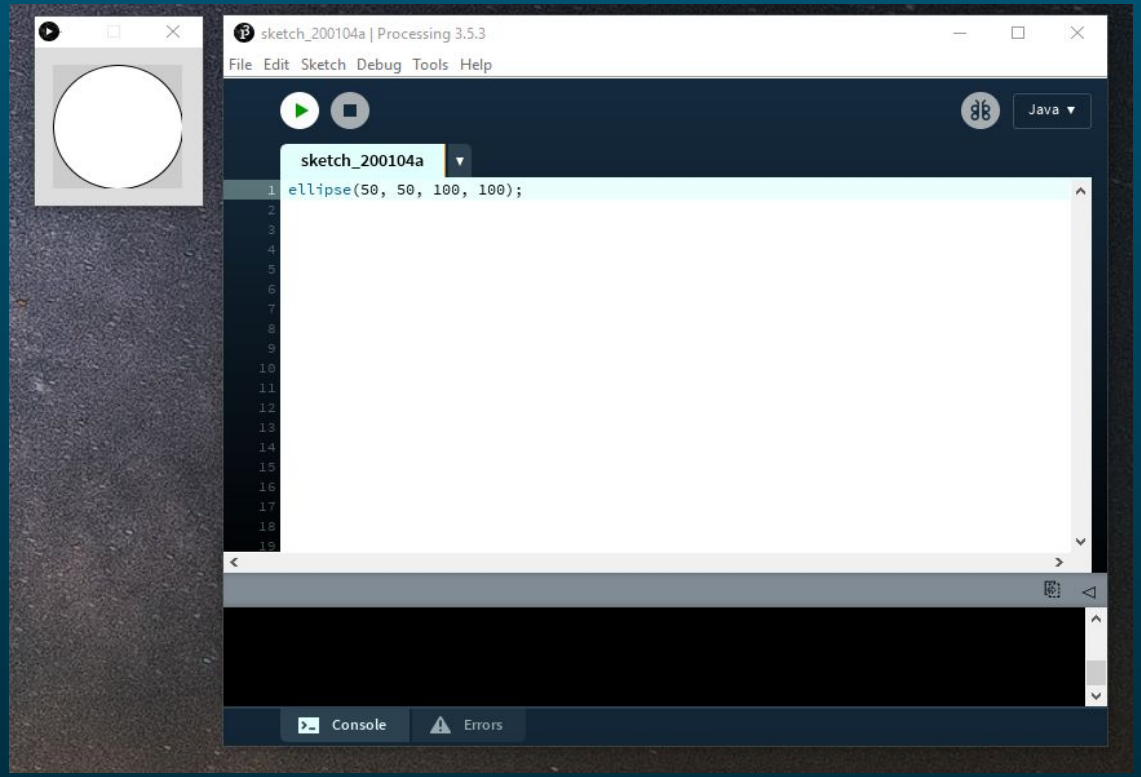
```
/**
 * Letters.
 *
 * Draws letters to the screen. This requires loading a font,
 * setting the font, and then drawing the letters.
 */

PFo
```

Cut
Copy
Copy as HTML
Paste
Select All
Comment/Uncomment
→ Increase Indent
← Decrease Indent
Find in Reference
Show Usage...
Jump to Declaration
Rename...

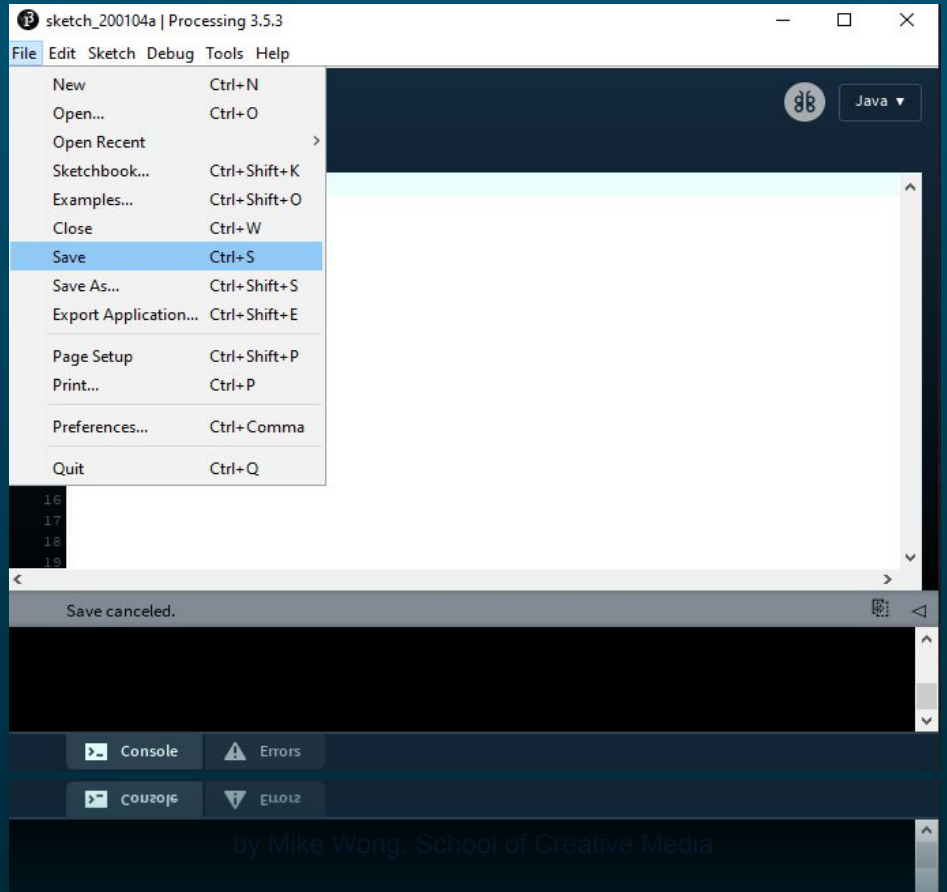
Processing Development Env. (PDE)



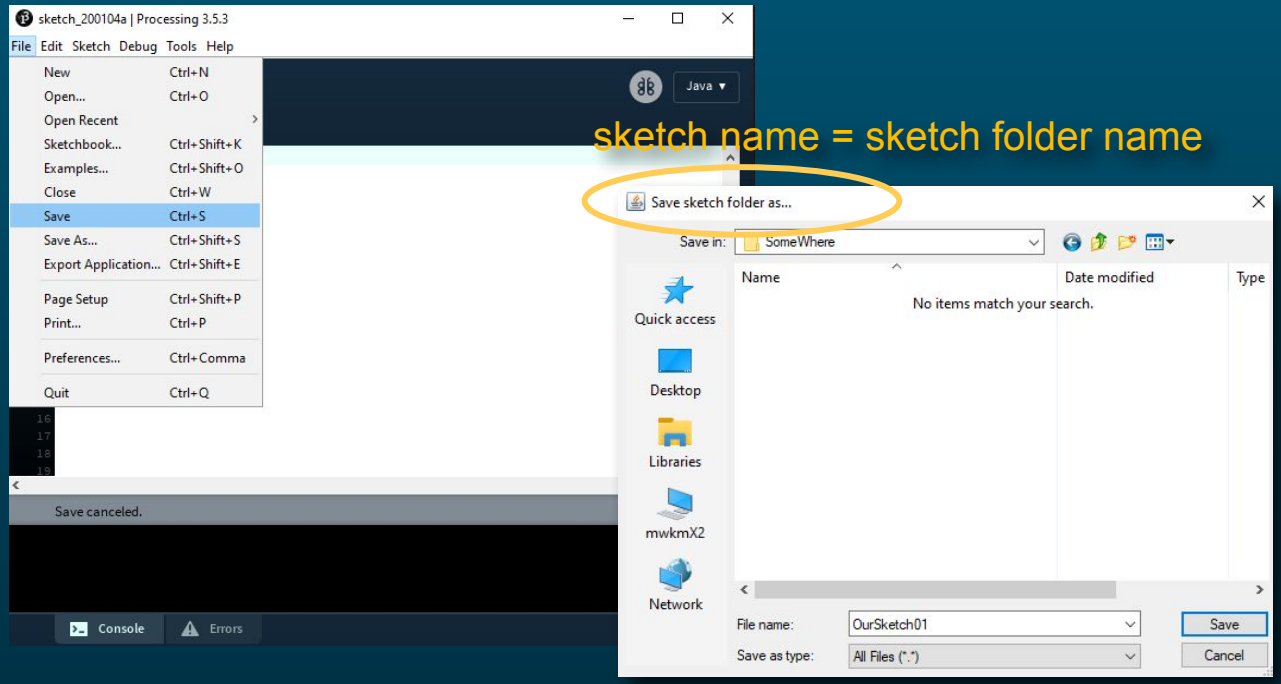
Our First Processing Sketch



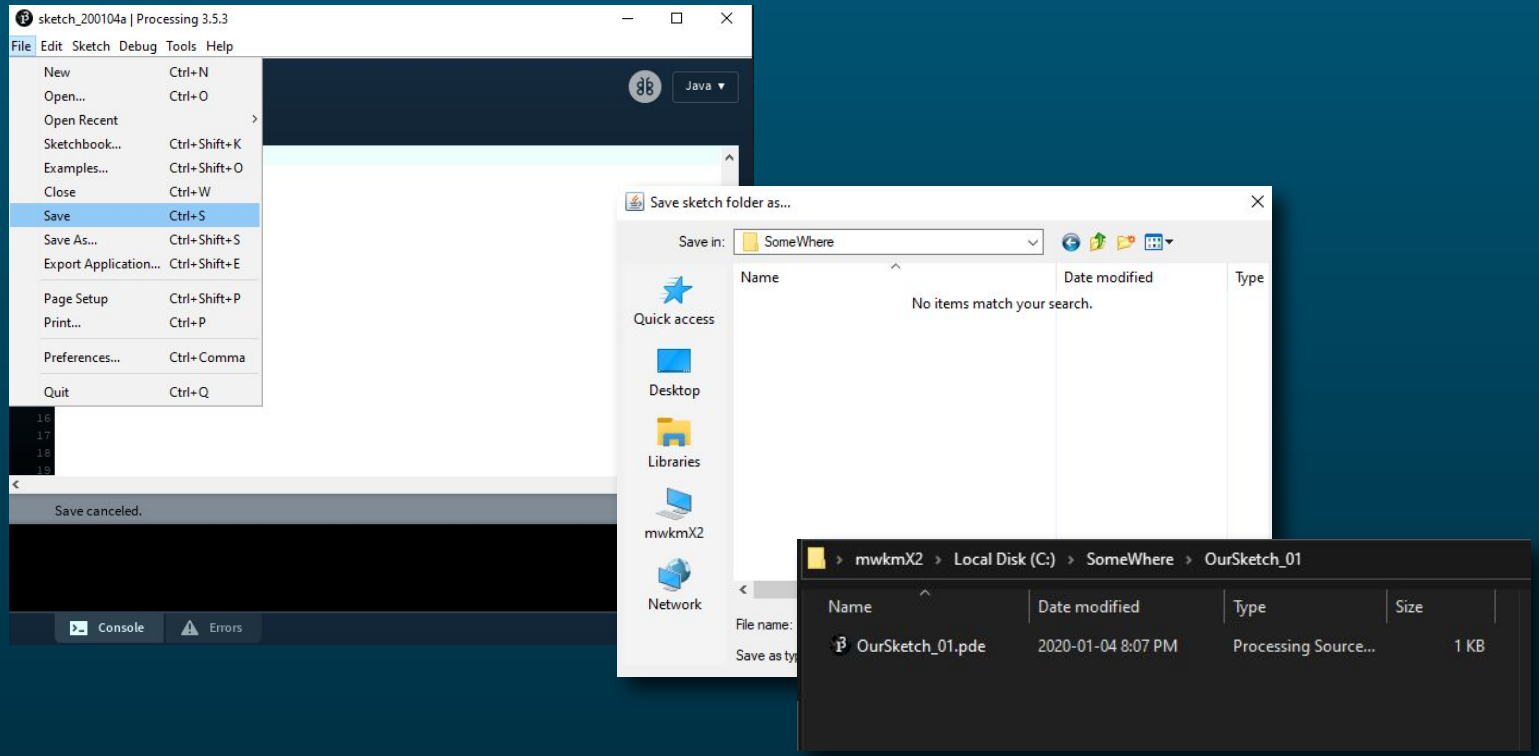
To save our sketch



To save our sketch

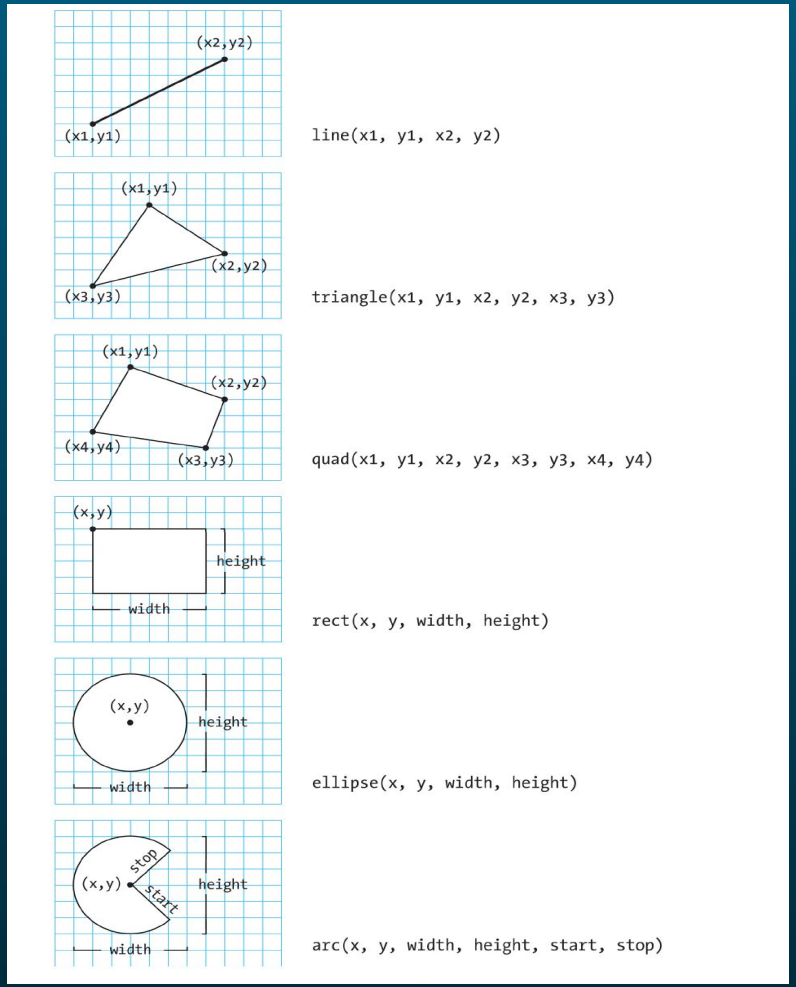


To save our sketch



sketch name = sketch folder name = sketch .pde file name

2D PRIMITIVE SHAPES AND THEIR COORDINATES



From p5.js to Processing

p5.js

```
let yPos = 0;
function setup() {
  createCanvas(200, 400);
}
function draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

From p5.js to Processing

p5.js

```
let yPos = 0;
function setup() {
  createCanvas(200, 400);
}
function draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

Processing

```
int yPos = 0;
void setup() {
  size(200, 400);
}
void draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

From p5.js to Processing

p5.js

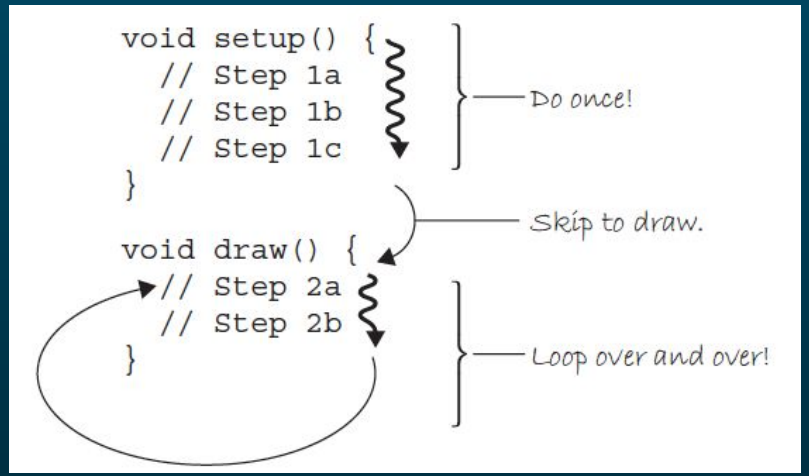
```
let yPos = 0;
function setup() {
  createCanvas(200, 400);
}
function draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

Processing

```
int yPos = 0;
void setup() {
  size(200, 400);
}
void draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

Structure & Flow of Program

- `void setup()`: called first and once, to define initial environment properties
- `void draw()`: continuously executes the lines inside until the program is stopped or when `noLoop()` is called
- `noLoop()`: stops Processing from continuously executing the code within `draw()` (to resume, use `loop()`)



Unstructured Program

- Does not contain `setup()` nor `draw()` functions
- Static sketch only - no interaction, dynamics and animation

```
background(0);          // Set the black background
stroke(255);            // Set line value to white
strokeWeight(5);        // Set line width to 5 pixels
smooth();               // Smooth line edges
line(10, 80, 30, 40);   // Left line
line(20, 80, 40, 40);
line(30, 80, 50, 40);   // Middle line
line(40, 80, 60, 40);
line(50, 80, 70, 40);   // Right line
```

0-02

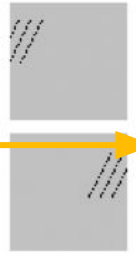
```
int x = 5;    // Set the horizontal position
int y = 60;   // Set the vertical position
line(x, y, x+20, y-40);     // Line from [5,60] to [25,20]
line(x+10, y, x+30, y-40); // Line from [15,60] to [35,20]
line(x+20, y, x+40, y-40); // Line from [25,60] to [45,20]
line(x+30, y, x+50, y-40); // Line from [35,60] to [55,20]
line(x+40, y, x+60, y-40); // Line from [45,60] to [65,20]
```

0-03

Structured Program

- Contains one `setup()` and one `draw()` function
- Dynamic sketches - allow animation and interactivity

`size(w, h);`
it is always the first line
in `setup()`



```
int x = 0;   // Set the horizontal position                    0-04
int y = 55;  // Set the vertical position

void setup() {
  size(100, 100);  // Set the window to 100 x 100 pixels
}

void draw() {
  background(204);
  line(x, y, x+20, y-40);      // Left line
  line(x+10, y, x+30, y-40);   // Middle line
  line(x+20, y, x+40, y-40);   // Right line
  x = x + 1;       // Add 1 to x
  if (x > 100) {   // If x is greater than 100,
    x = -40;       // assign -40 to x
  }
}
```

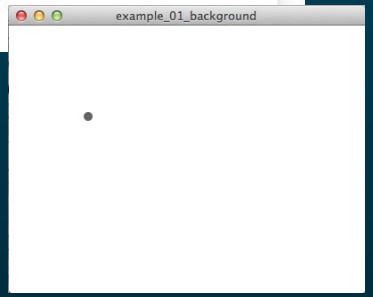
Example 1

Where to put `background(255);` ?

Compare these two programs:

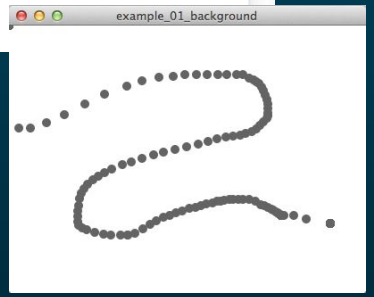
example_01_background

```
void setup() {
  size(400, 300);
  noStroke();
  fill(100);
}

void draw() {
  background(255);
  ellipse(mouseX, mouseY, 10, 10);
}
```



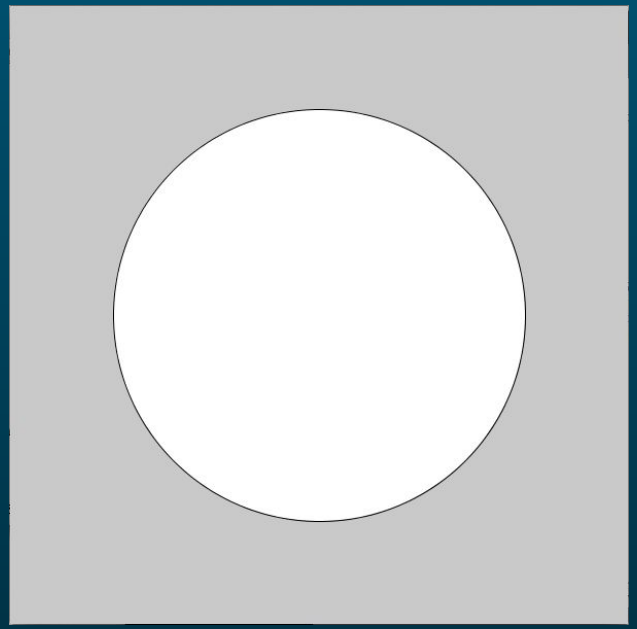
example_01_background

```
void setup() {
  size(400, 300);
  background(255);
  noStroke();
  fill(100);
}

void draw() {
  ellipse(mouseX, mouseY, 10, 10);
}
```



In-class Exercise 1 (Canvas)

- Write a structured program
- Use a display window of dimension 600 x 600
- The background color has to change from white to black, then white to black again continuously
- Draws a circle of radius 200 at the middle of the canvas



Exercise 1 reference code

- Background color `bg` is updated in every frame, thus `background(bg)` has to be put inside `void draw()`, not in `void setup()`
- `bg = bg - 1;` background color `bg` is decreased by 1 in each iteration of `draw()`, i.e. `255 -> 254 -> 253 ... 2 -> 1 -> 0`
- When background color `bg` reached a value of `-1`, we reset it to `255` (white) again
- The dimension of the canvas is 600x600, thus centre of the canvas is `(300,300)`
- Although the circle is static (not changing), it has to be drawn each time after the canvas has been cleared with the background color `bg`.

w01_exercise01 | Processing 3.5.3

File Edit Sketch Debug Tools Help

Java

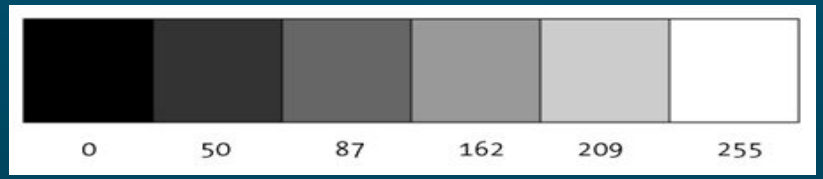
w01_exercise01

```
//  SM2715 Creative Coding L02
//  by Mike Wong

//  Variable declaration
int bg = 255;  // use an 'int'eger variable to store our grayscale color.

//  Processing Starting Point
void setup() {
  size(600,600);
}

//  Continuous execution by Processing
void draw() {

  //  Clear whole canvas with color 'bg'
  background(bg);

  //  Decrease bg  by 1
  bg = bg - 1;
  //  Check if bg is within range of 0-255
  if (bg < 0) {
    bg = 255;
  }

  //  Draw our static ellipse
  ellipse(300,300,400,400);

}
```

Color and Data Types In Processing

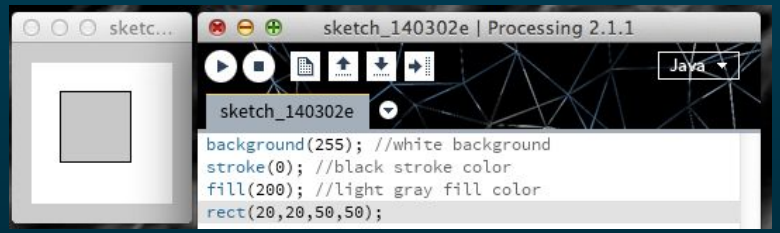
Grayscale Color

- 8-bit: 0-255 (2^8 = 256)
- 0 as black and 255 as white



- e.g.

```
background(255); // white background
stroke(0); // black stroke color
fill(200); // light gray fill color
```



Color

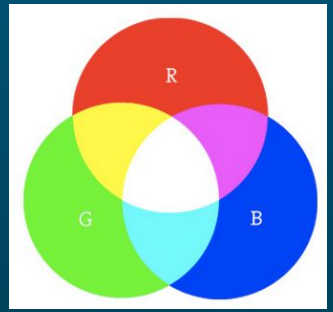
Setting
- background()
- clear()
- colorMode()
- fill()
- noFill()
- noStroke()
- stroke()

Creating & Reading
- alpha()
- blue()
- brightness()
- color()
- green()
- hue()
- lerpColor()
- red()
- saturation()

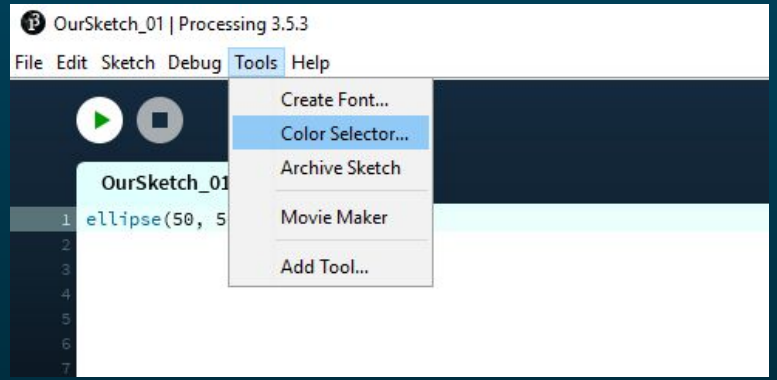
RGB Color

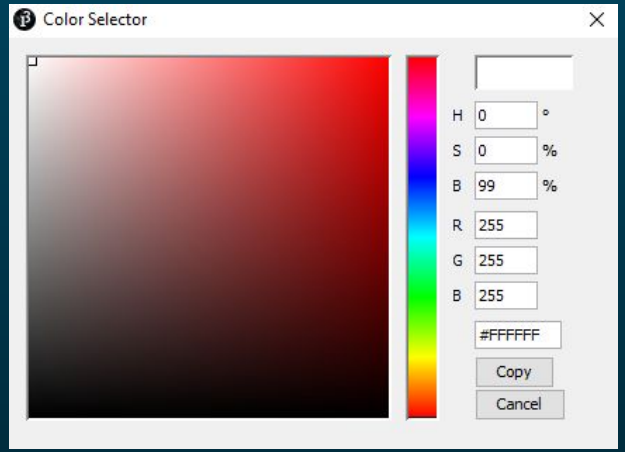
- 24-bit full color
- 8-bit per-channel of red, green and blue (8-bit x 3)
- Examples:

```
background(255,0,0); // red
fill(48,139,206);
```



- Color selector tool in Processing PDE

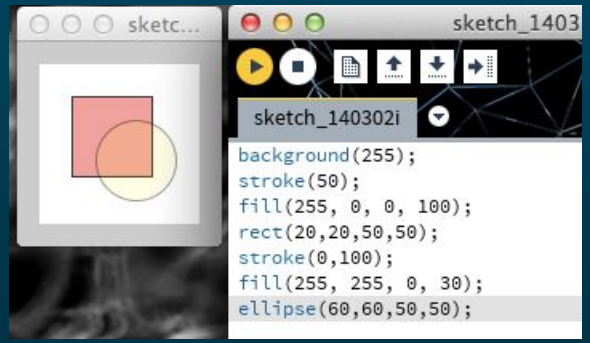




Transparency (RGBAlpha Value)

- Alpha values range from `0` to `255`
- `0`: completely transparent (i.e., 0% opaque)
- `255`: completely opaque (i.e., 100% opaque).
- e.g.

```
fill(0,50);
stroke(255,0,0,100);
```



- Transparency is <u>not applicable</u> to background !

Example: Alpha transparency

```
size(200,200);
background(0);
noStroke();

// No fourth argument means 100% opacity.
fill(0,0,255);
rect(0,0,100,200);

// 255 means 100% opacity.
fill(255,0,0,255);
rect(0,0,200,40);

// 75% opacity.
fill(255,0,0,191);
rect(0,50,200,40);

// 55% opacity.
fill(255,0,0,127);
rect(0,100,200,40);

// 25% opacity.
fill(255,0,0,63);
rect(0,150,200,40);
```

Variable Declaration & Initialization

- **Declaration rule:** `<data_type> <variable_name> = [initial_value];`
  - `int a = 10;`

- **Variable naming rules**:
  - Must be one word (no space)
  - Must start with a letter
    - Can include numbers; but not start with a number
  - Can include underscore "_"
  - Must be unique
    - CANNOT use the name of system variables, e.g.,
      - `mouseX`, `mouseY`, `width`, `height`, `frameCount`, `key`, `keyCode`, `keyPressed`, `mousePressed`, `mouseButton` etc.

Data Types

Name	Value range	Example
boolean	true or false	boolean T = true, F = false;
char	0 to 65535 (16-bit)	char myCh1='A', myCh2='$';
byte	-128 to 127 (8-bit)	byte b = -128;
int	-2147483648 to 2147483647 (32-bit integer)	int number = 800, temp = -1;
float	3.40282347E+38 to -3.40282347E+38(32 bits)	float b = -2.984;
color	16,777,216 colors RGBA 32-bit (24-bit + 8-bit)	color c1 = color(204,153,0), c2 = #FFCC00;

Data Types

```
int x;        // Declare a variable named x of data type int
float y;      // Declare a variable named y of data type float
boolean b;    // Declare a variable named b of data type boolean
x = 50;       // the variable x is assigned a value of 50
y = 12.6;     // the variable y is assigned a value of 12.6
b = true;     // the variable b is assigned a value of true
```

Data Type Conversion*

Data
Primitive
boolean
byte
char
color
double
float
int
long

Conversion
binary()
boolean()
byte()
char()
float()
hex()
int()
str()
unbinary()
unhex()

```
float f = 65.0;
int i = int(f);
println(f + " : " + i);  // Prints "65.0 : 65"

char c = 'E';
i = int(c);
println(c + " : " + i);  // Prints "E : 69"
```

Data Type Conversion*

Data
Primitive
boolean
byte
char
color
double
float
int
long

Conversion
binary()
boolean()
byte()
char()
float()
hex()
int()
str()
unbinary()
unhex()

```
boolean b = false;
byte y = -28;
char c = 'R';
float f = -32.6;
int i = 1024;

String sb = str(b);
String sy = str(y);
String sc = str(c);
String sf = str(f);
String si = str(i);

sb = sb + sy + sc + sf + si;

println(sb);  // Prints 'false-28R-32.61024'
```

ASCII Table

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

Example 2

w01_code_02 | Processing 3.5.3

File Edit Sketch Debug Tools Help

Java

w01_code_02

```
boolean a = false;
char b = '1';
char c = 97;
byte d = -128;
int e = 2047;
float f = 3.1415;
float g = 3;
color h = color(204, 153, 0);
color i = #FFCC00;

println(a);
println(b);
println(c);
println(d);
println(e);
println(f);
println(g);
println(h);
println(i);
```

```
false
1
a
-128
2047
3.1415
3.0
-3368704
-13312
```

Console Errors