

WEEK 3

IMAGE PROCESSING

IMAGE LOADING

- **PImage**

Datatype for storing images

- Syntax

`PImage imageName; // Declare a variable type PImage`

- To add an image file to the “data” directory

Sketch → Add File... or

Sketch → Show Sketch Folder, then create a data directory and place the image file inside

- To load an image into a type PImage variable

`loadImage(filename)`

eg. `imageName = loadImage("filename.jpg");`

Fields and Methods of PImage

Fields

`pixels[]` Array containing the color of every pixel in the image

`width` The width of the image in units of pixels

`height` The height of the image in units of pixels

Methods

`loadPixels()` Loads the pixel data for the image into its `pixels[]` array

`updatePixels()` Updates the image with the data in its `pixels[]` array

`resize()` Resize the image to a new width and height

`get()` Reads the color of any pixel or grabs a rectangle of pixels

`set()` Writes a color to any pixel or writes an image into another

`mask()` Masks part of an image with another image as an alpha channel

`filter()` Converts the image to grayscale or black and white

`copy()` Copies the entire image

`blendColor()` Blends two color values together based on the blending mode given as the `MODE` parameter

`blend()` Copies a pixel or rectangle of pixels using different blending modes

`save()` Saves the image to a TIFF, TARGA, PNG, or JPEG file

Use the Dot operator to access the fields or methods of a PImage instance:

`imageName.fieldName`

`imageName.methodName`

DISPLAYING IMAGE

- Fields of a PImage variable:

`pixels[], width, height`

eg `myImg.pixels[i], myImg.width, myImg.height`

- To display an image

`image(myImg, x, y);`

`// image displayed at its original size with its`

`// upper-left corner at coordinates (x,y)`

or

`image(myImg, x, y, w, h);`

`// image displayed at size (w x h)`

`// with its upper-left corner at coordinates (x,y)`

EXAMPLE 1 LOAD AND DISPLAY IMAGE

```
PImage img = loadImage("Haeckel01.jpg");  
size(450,450);  
image(img, 0, 0);
```

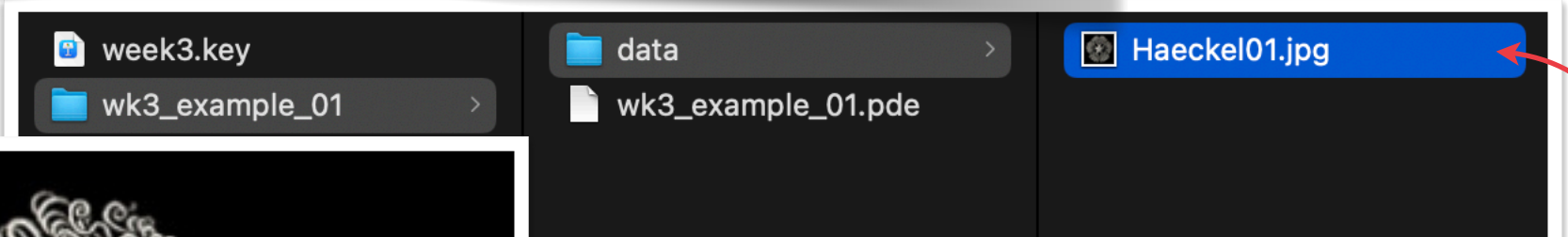


Image file should be placed in "data" folder

DISPLAYING IMAGE

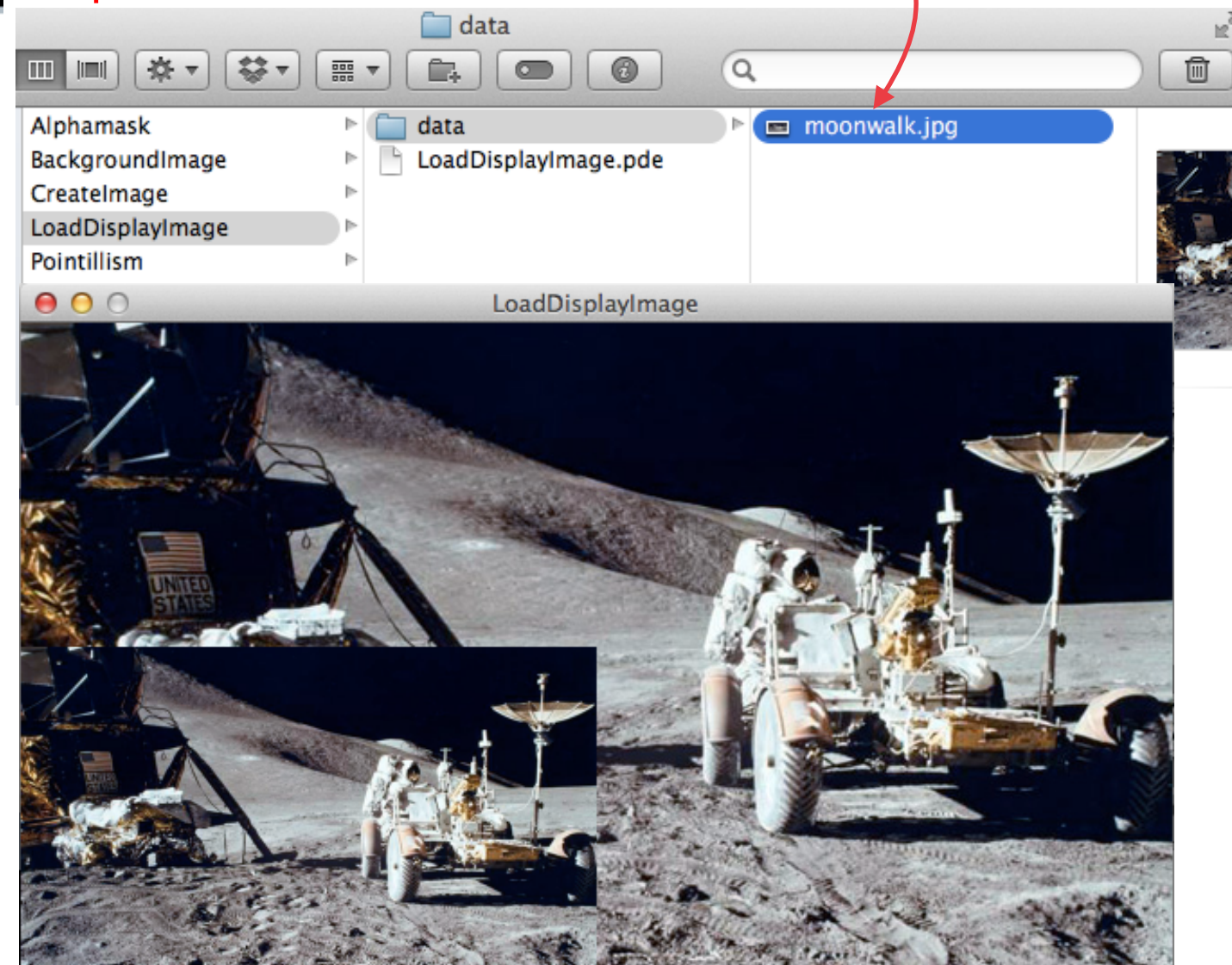
```
PImage imageName; // Declare  
imageName = loadImage("filename.jpg"); // Load  
image(imageName, 0, 0); // Display
```

Add Examples...

- Basics
 - Arrays
 - Camera
 - Color
 - Control
 - Data
 - Form
 - Image
 - Alphamask
 - BackgroundImage
 - CreateImage
 - LoadDisplayImage

Image file should be placed in "data" folder

```
LoadDisplayImage | Processing 2.0.3  
Java  
LoadDisplayImage  
/**  
 * Load and Display  
 *  
 * Images can be loaded and displayed to the screen at their actual size  
 * or any other size.  
 */  
  
// The next line is needed if running in JavaScript Mode with Processing.js  
/* @pjs preload="moonwalk.jpg"; */  
  
PImage img; // Declare variable "a" of type PImage  
  
void setup() {  
  size(640, 360);  
  // The image file must be in the data folder of the current sketch  
  // to load successfully  
  img = loadImage("moonwalk.jpg"); // Load the image into the program  
}  
  
void draw() {  
  // Displays the image at its actual size at point (0,0)  
  image(img, 0, 0);  
  // Displays the image at point (0, height/2) at half of its size  
  image(img, 0, height/2, img.width/2, img.height/2);  
}
```



EXAMPLE 2 LOADING IMAGE FROM WEB

```
PImage webImg;
```

```
void setup() {  
  String url = "http://totallyhistory.com/wp-content/uploads/2012/12/the-birthday-1915-chagall.jpg";  
  // Load image from web  
  //webImg = loadImage(url);  
  
  // requestImage() runs on a separate thread so that  
  // your sketch doesn't freeze while images load during setup()  
  webImg = requestImage(url);  
  size(939, 761);  
}  
  
void draw() {  
  background(0);  
  ellipse(mouseX, mouseY, 200, 200);  
  image(webImg, 0, 0);  
}
```



Marc Chagall

IMAGE FORMATS

- Processing can display GIF, JPG, and PNG images

Format	Extension	Color depth	Transparency
GIF	.gif	1-bit to 8-bit	1-bit
JPEG	.jpg	24-bit	None
PNG	.png	1-bit to 24-bit	8-bit

IMAGEMODE

- `image(imageName, a, b)` or `image(imageName, a, b, c, d)`

- Syntax

`imageMode(CORNER)` - default, `(a, b)` defines the upper-left corner of the image

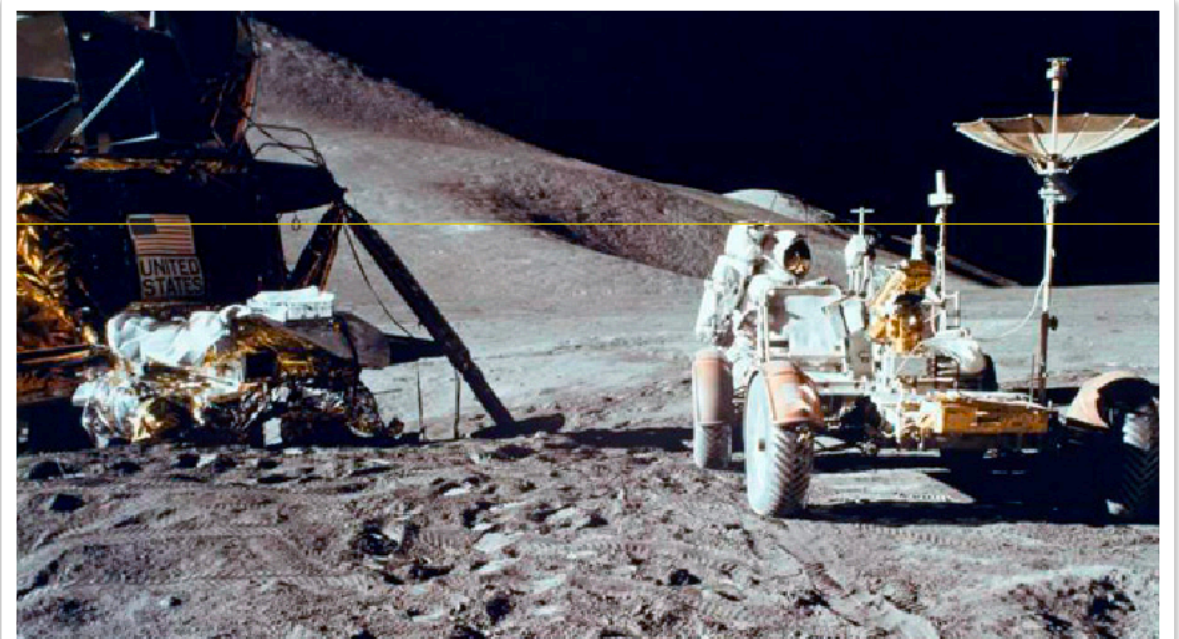
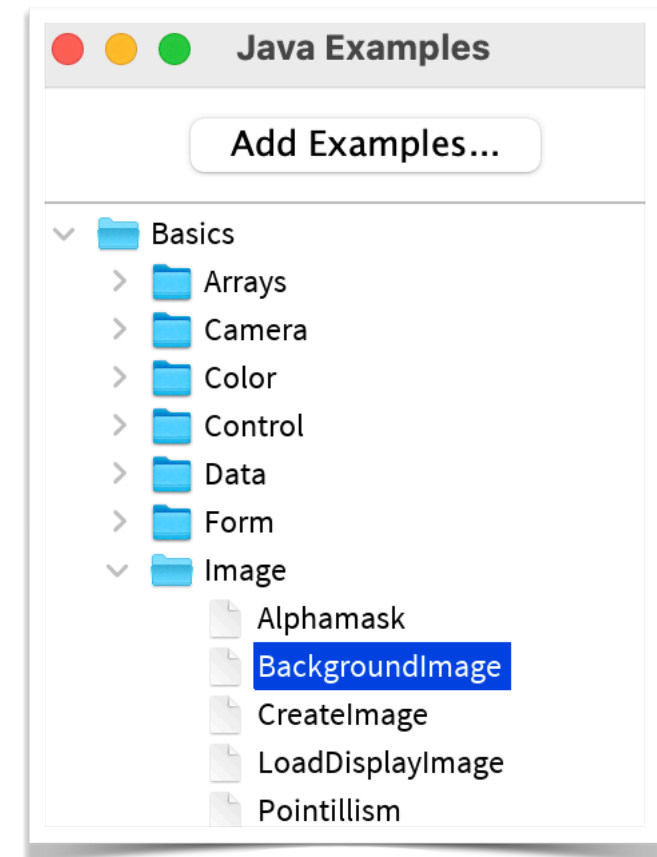
`imageMode(CORNERS)` - `(a, b)` defines the upper-left corner of the image, while `(c, d)` defines the opposite corner

`imageMode(CENTER)` - `(a, b)` defines the center of the image, while `(c, d)` (if exist) defines the width and height of the image

BACKGROUND IMAGE

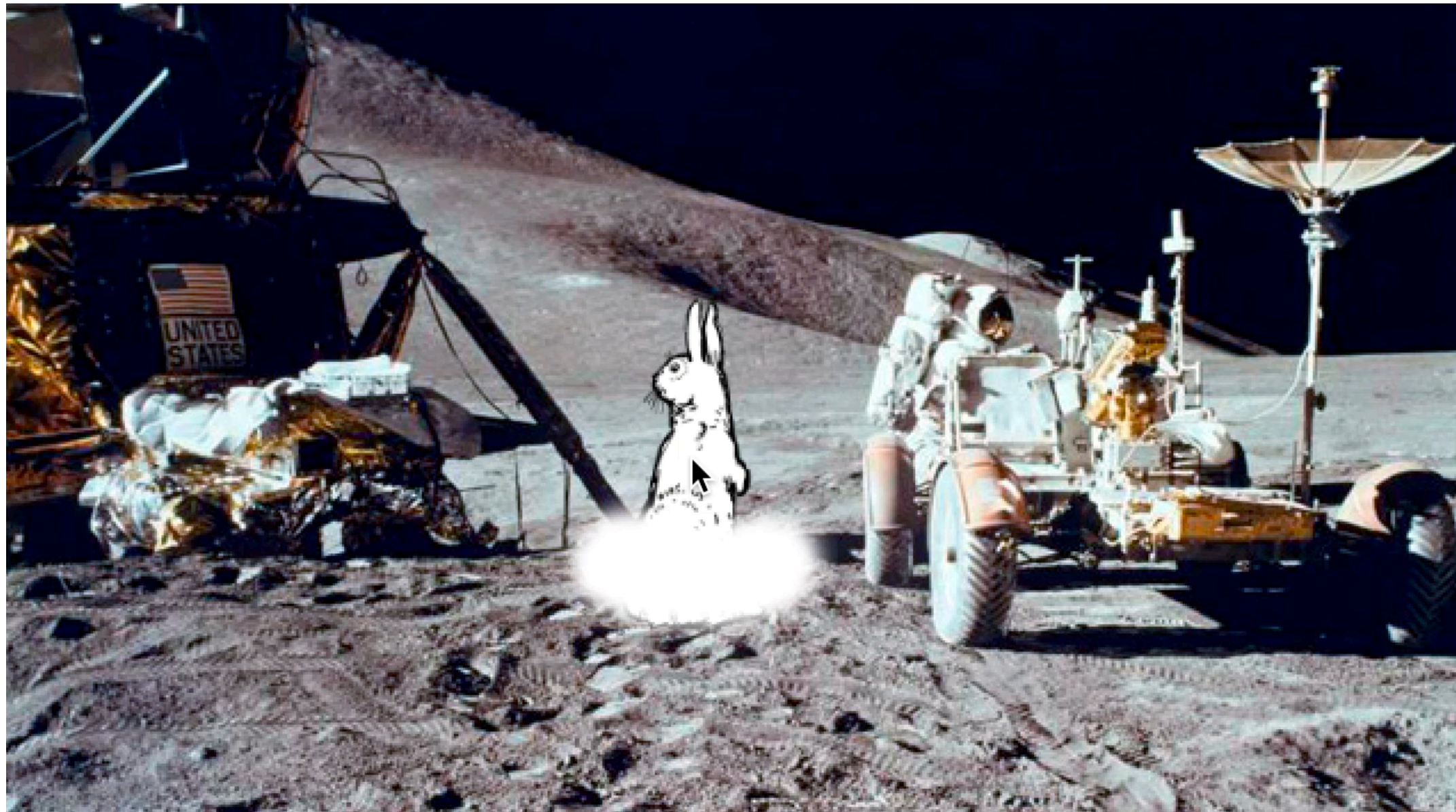
Example Basics/Image/BackgroundImage

```
PImage bg;
int y;
void setup() {
  size(640, 360);
  bg = loadImage("moonwalk.jpg");
}
void draw() {
  background(bg);
  stroke(226, 204, 0);
  line(0, y, width, y);
  y++;
  if (y > height) {
    y = 0;
  }
}
```



EXERCISE 1

- Based on Example Basics/Image/BackgroundImage, display moonwalk.jpg as background
- Display another image in proper size at the mouse position



EXAMPLE 3 GET() METHOD

- Syntax

`img.get(x, y)` - getting the color of a single pixel

`img.get(x, y, w, h)` - getting a section of the display window by specifying an additional width and height parameter

`img.get()` - getting the entire image

```
PImage img;  
void setup() {  
  size(640, 360);  
  img = loadImage("moonwalk.jpg");  
  noStroke();  
  rectMode(CENTER);  
}  
void draw() {  
  image(img, 0, 0);  
  fill(img.get(mouseX, mouseY));  
  rect(mouseX, mouseY, 50, 50);  
}
```

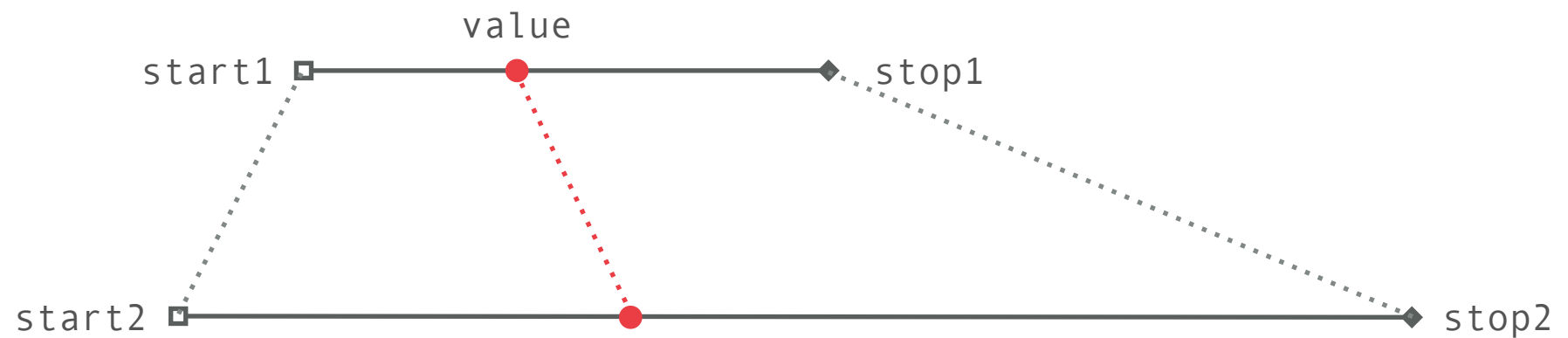


MAPPING

- Re-maps a number from one range to another.

- Syntax

```
map(value, start1, stop1, start2, stop2)
```

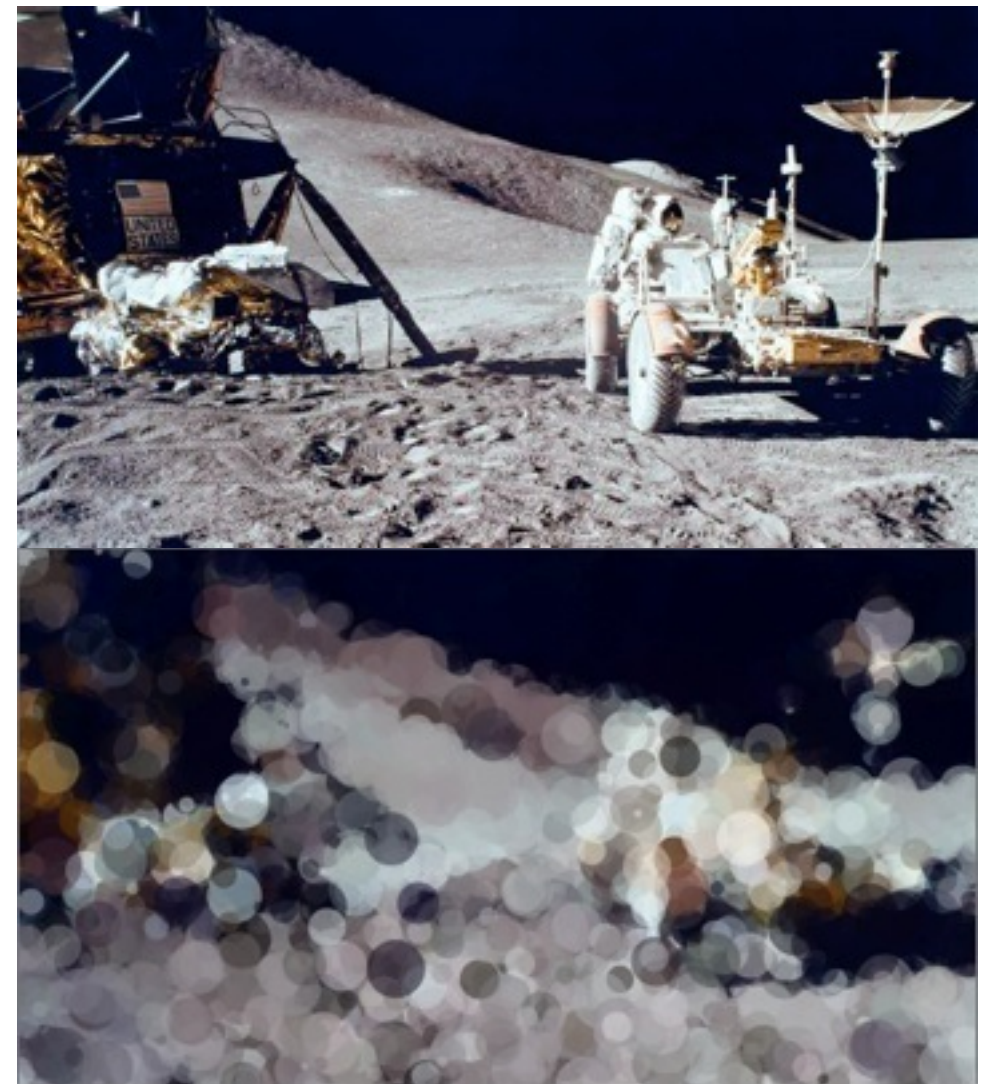


- Example

```
void draw() {  
    float x = map(mouseX, 0, width, 0, 255);  
    background(x);  
    println(int(x));  
}
```

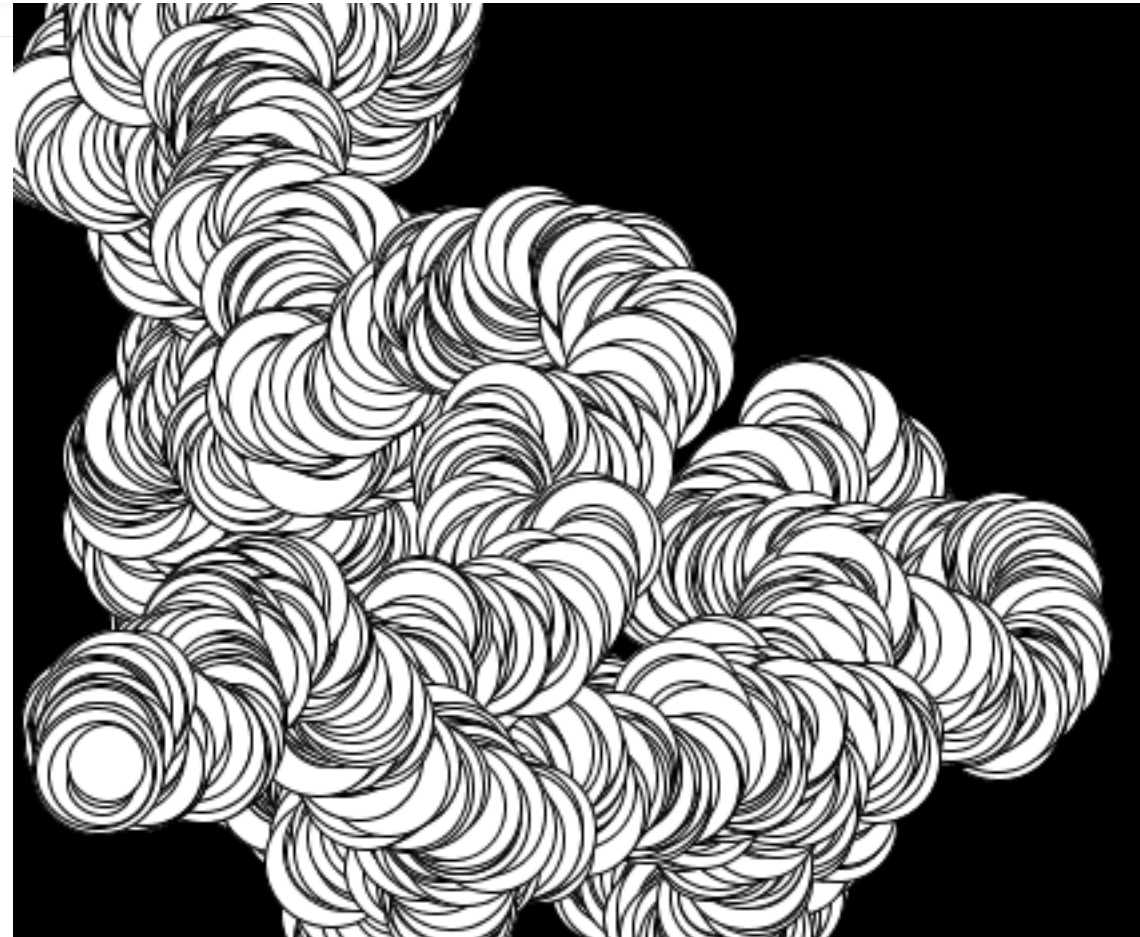
EXAMPLE Basics/Image/Pointillism

```
PImage img;  
int smallPoint, largePoint;  
void setup() {  
  size(640, 360);  
  img = loadImage("moonwalk.jpg");  
  smallPoint = 4;  
  largePoint = 40;  
  imageMode(CENTER);  
  noStroke();  
  background(255);  
}  
void draw() {  
  float pointillize = map(mouseX, 0, width, smallPoint, largePoint);  
  int x = int(random(img.width));  
  int y = int(random(img.height));  
  color pix = img.get(x, y);  
  fill(pix, 128);  
  ellipse(x, y, pointillize, pointillize);  
}
```



EXAMPLE 4 RANDOM WALK

```
PImage img;  
int x, y;  
int speed = 10;  
float diameter;  
void setup() {  
  size(640, 360);  
  x = width/2;  
  y = height/2;  
  img = loadImage("moonwalk.jpg");  
  background(0);  
  noStroke();  
}  
void draw() {  
  x += (int)random(-speed, speed);  
  y += (int)random(-speed, speed);  
  x = constrain(x, 0, width-1);  
  y = constrain(y, 0, height-1);  
  diameter = random(5, 30);  
  fill(img.get(x, y), 50);  
  ellipse(x, y, diameter, diameter);  
}
```



TINT: IMAGE COLOR & TRANSPARENCY

- Sets the fill value for displaying images. Images can be tinted to specified colors or made transparent
- Syntax (affects only images)

`tint(rgb)`

`tint(rgb, alpha)`

`tint(gray)`

`tint(gray, alpha)`

`tint(v1, v2, v3)`

`tint(v1, v2, v3, alpha)`

```
// Example 5
```

```
// Based on Daniel Shiffman
```

```
// Learning Processing 2nd Edition P.307
```

```
PImage sunflower =  
loadImage("sunflower.jpg");
```

```
PImage dog = loadImage("dog.jpg");
```

```
size(200, 200);
```

```
background(dog);
```

```
tint(255); ← apply tint()  
//tint(100); // before image()
```

```
//tint(255, 127);
```

```
//tint(0, 200, 255);
```

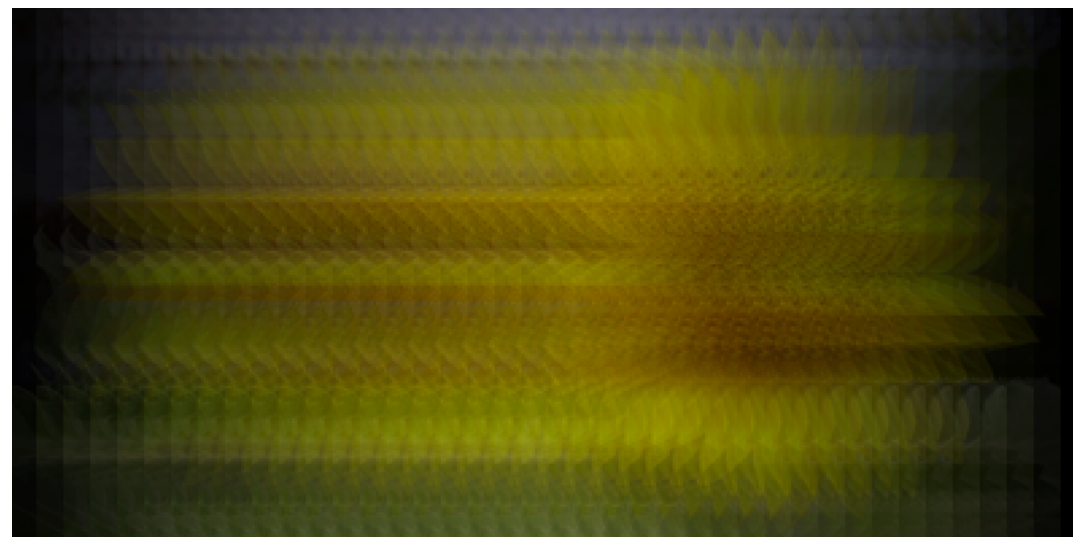
```
//tint(255, 0, 0, 100);
```

```
image(sunflower, 0, 0);
```



EXAMPLE 6

```
// Based on Daniel Shiffman
// Learning Processing 2nd Edition P.307
// and Processing Handbook Example 10-07
PImage sunflower = loadImage("sunflower.jpg");
size(400, 200);
background(0);
tint(255, 20);
// Draw the image 20 times, moving each to the right
for (int i = 0; i < 20; i++)
    image(sunflower, i*10, 0);
```



PRESET IMAGE FILTERS

Processing offers a series of preset filters that can be applied to the display window:

Syntax

```
filter(mode)
```

```
filter(mode, range)
```

```
filter(shader)
```

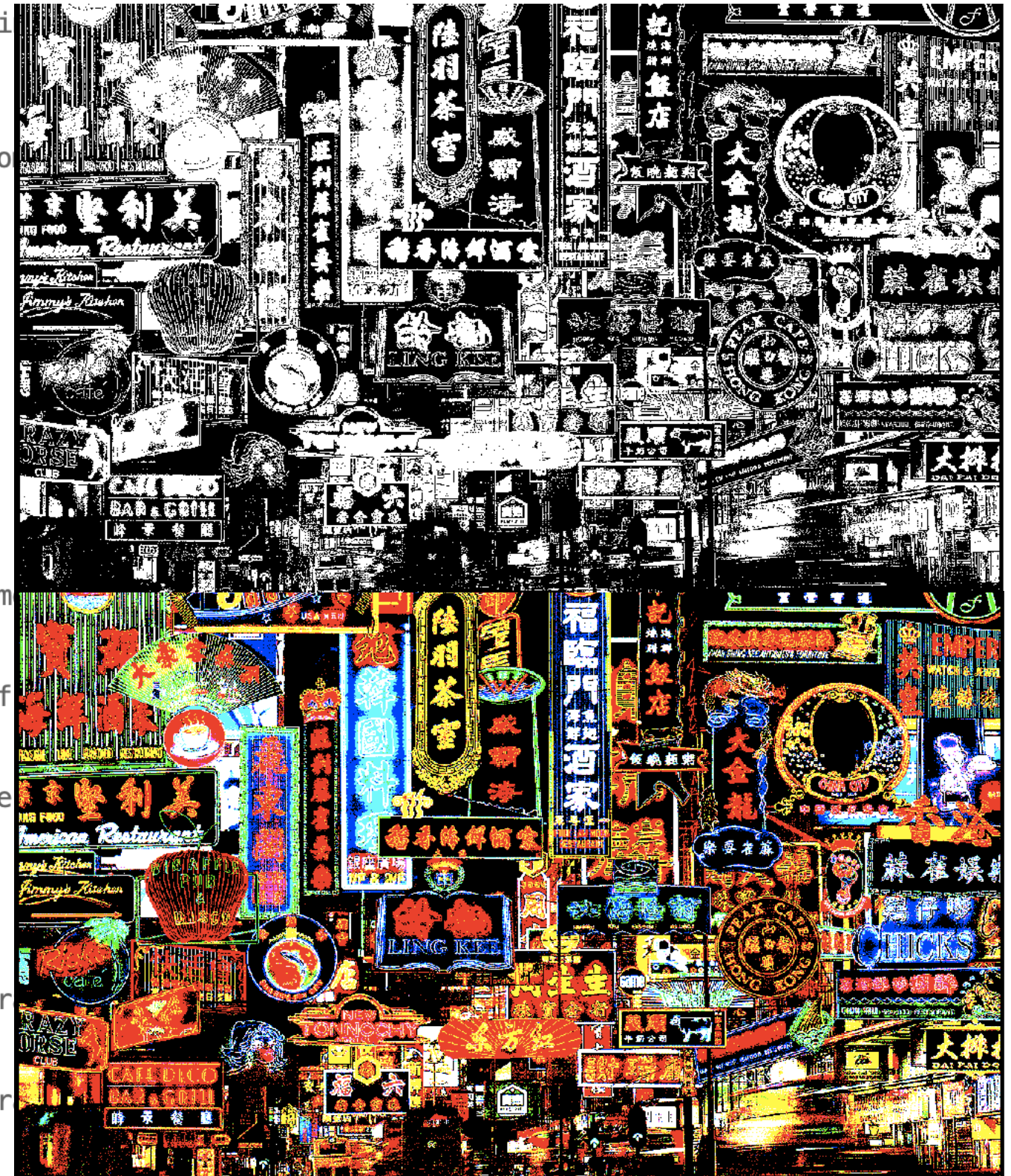
Mode

```
THRESHOLD(range 0-1), GRAY, OPAQUE, INVERT,  
POSTERIZE(range 2-255), BLUR(range >=1), ERODE or  
DILATE
```

Apply `filter()` **after** `image()`

EXAMPLE 7

```
1 // Based on Algorithms for Visual Design Using the Processing
2 PImage myImage; //define an image object
3
4 void setup() {
5   myImage = loadImage("neon.jpg"); //load image - photo from
6   size(1280, 938);
7 }
8
9 void draw() {
10  image(myImage, 0, 0); //display the image
11  //if (keyPressed==true)
12  if (key>='0' && key<='8') {
13    switch(key) {
14      case '0':
15        filter(OPAQUE); // resets image
16        break;
17      case '1':
18        filter(THRESHOLD, .6); //every pixel below .6 becomes
19        break;
20      case '2':
21        filter(GRAY); //all pixels get the average value of
22        break;
23      case '3':
24        filter(INVERT); //all pixels get the opposite value
25        break; // (i.e. 255-r)
26      case '4':
27        filter(POSTERIZE, 2); //limits each channel of the
28        break; // colors
29      case '5':
30        filter(BLUR, 1); // executes a Guassian blur with r
31        break;
32      case '6':
33        filter(BLUR, 6); // executes a Guassian blur with r
34        break;
35      case '7':
```



Fields and Methods of PImage

Fields	<code>pixels[]</code>	Array containing the color of every pixel in the image
	<code>width</code>	Image width
	<code>height</code>	Image height
Methods	<code>loadPixels()</code>	Loads the pixel data for the image into its <code>pixels[]</code> array
	<code>updatePixels()</code>	Updates the image with the data in its <code>pixels[]</code> array
	<code>resize()</code>	Changes the size of an image to a new width and height
	<code>get()</code>	Reads the color of any pixel or grabs a rectangle of pixels
	<code>set()</code>	writes a color to any pixel or writes an image into another
	<code>mask()</code>	Masks part of an image with another image as an alpha channel
	<code>filter()</code>	Converts the image to grayscale or black and white
	<code>copy()</code>	Copies the entire image
	<code>blend()</code>	Copies a pixel or rectangle of pixels using different blending modes
	<code>save()</code>	Saves the image to a TIFF, TARGA, PNG, or JPEG file

EXAMPLE 8 filter() vs PImage.filter()

```
PImage img1, img2; //define an image object
```

```
void setup() {
```

```
  img1 = loadImage("baby.jpg");
```

```
  img2 = loadImage("baby.jpg");
```

```
  size(1200, 400);
```

```
  background(0);
```

```
  // apply filter to img1 only
```

```
  img1.filter(INVERT);
```

```
}
```

```
void draw() {
```

```
  image(img1, 0, 0);
```

```
  image(img2, 400, 0);
```

```
  // apply filter to the entire display window
```

```
  filter(GRAY);
```

```
  image(img2, 800, 0);
```

```
}
```

