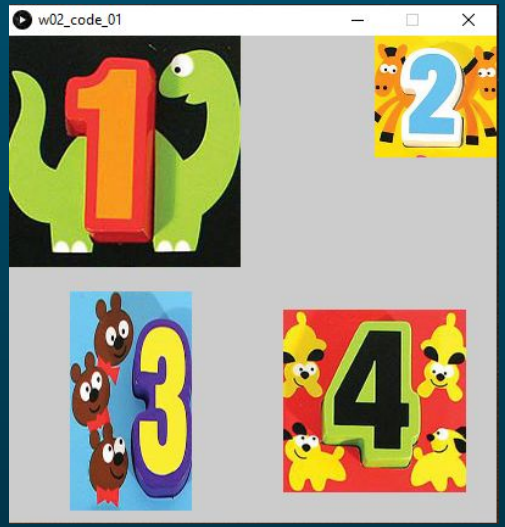
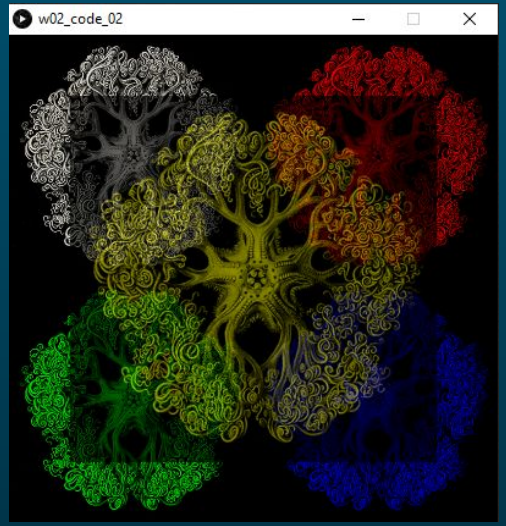
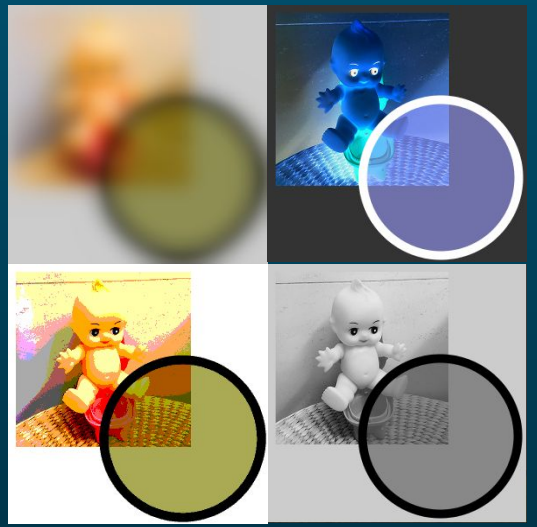
Week 02
Playing with Image Part 1

by Mike Wong. School of Creative Media

Playing with images in Processing







PImage Image datatype in Processing

PImage : datatype for working with Image

Example

```
// Declare a variable of type PImage
PImage image0;
image0 = loadImage("picture01.jpg");
```

loadImage(<file>)

loadImage(<file>) : loads an external image to a PImage typed variable.

Example

```
PImage image0;
// load an image file into image0
image0 = loadImage("picture01.jpg");
```

loadImage(<file>)

```
PImage image0;
image0 = loadImage("picture01.jpg");
```

Processing always assumes the image file `picture01.jpg` is available in a folder named **`data` inside our sketch folder**. An example here:

```
C:\mysketch\mysketch.pde
C:\mysketch\data\picture01.jpg
```

loadImage(<URL>)

loadImage(<URL>) : loads an image from a URL from a web to a PImage typed variable.

Example

```
PImage image0;
image0 = loadImage("https://art.github.io/cc_2019B/images/Haeckel01.jpg");
```

Supported Image Formats

Format	Extension	RGB 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

Display image with `image()`

```
image(<imageVar>, <x>, <y>, [w], [h]);
```

displays an image stored in **`imageVar`** (**`PImage`** typed) at position **`(x,y)`**, the upper left corner; with an option to scale the image to width and height defined by **`w`** & **`h`** respectively.

Display image with image()

```
image(<imageVar>, <x>, <y>, [w], [h]);
```

Example

```
PImage image0;
image0 = loadImage("picture1.jpg");
image(image0, 100, 100, 200, 200);
```

Image positioning with imageMode()

```
imageMode(<option>);
```

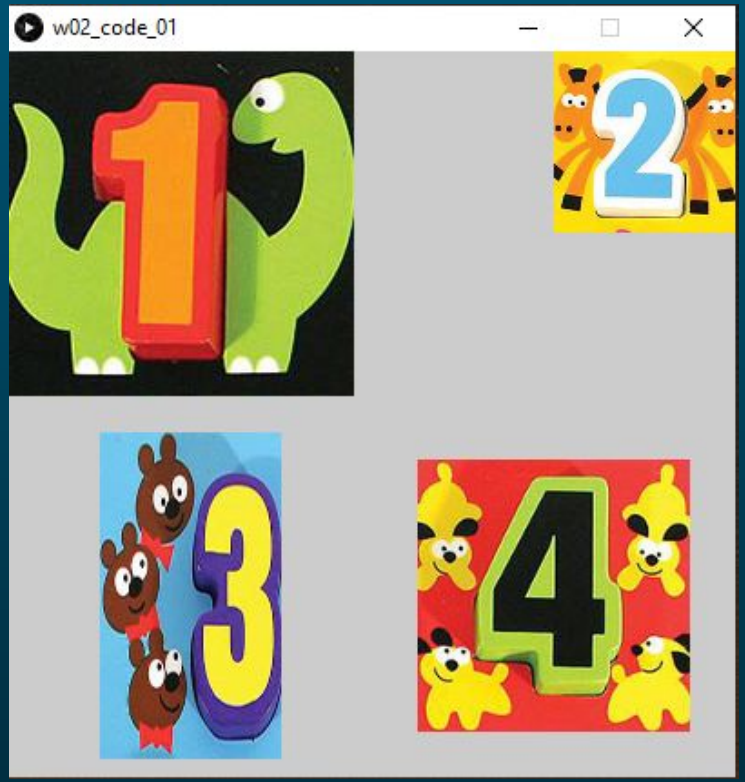
controls how the image is positioned by `image()`.

```
imageMode(CORNER);   // DEFAULT
imageMode(CORNERS);  // (w,h)-> LR corner
imageMode(CENTER);   // (x,y)-> center
```

Example 1

w02_code_01

```
PImage image1, image2, image3, image4;

void setup() {

  size(400, 400);
  //  Load the image files into our variables
  //  All images are of size 190 x 190 pixels

  //  LOCAL images (from sub-folder 'data')
  image1 = loadImage("1.jpg");
  image2 = loadImage("2.jpg");

  //  Images from URLs
  image3 = loadImage("https://artixels.github.io/cc_2019B/images/3.jpg");
  image4 = loadImage("https://artixels.github.io/cc_2019B/images/4.jpg");

  image(image1, 0, 0);
  image(image2, 300, 0, 100, 100);  // scaled

  imageMode(CORNERS);
  image(image3, 50, 210, 150, 390); // stretched

  imageMode(CENTER);
  image(image4, 300, 300, 150, 150);
}
```



Changing image display with `tint()`

Let us recall ….

```
// Shape drawing
fill(128);
stroke(255);
rect(0,0,100,100);
noFill();
```

Changing image display with tint()

Let us recall ….

```
// Shape drawing
fill(128);
stroke(255);
rect(0,0,100,100);
noFill();
```

```
// Image display
tint(128);
image(image0,0,0);
noTint();
```

Changing image display with tint()

```
tint(<GrayScale>);   tint(<R>,<G>,<B>);
tint(<GrayScale>,<Alpha>);
tint(<R>,<G>,<B>,<Alpha>);
```

changes the color and/or transparency of the forthcoming images to be displayed.

Example

```
tint(0,0,255);   // Bluish Tint
image(image0, 100, 100, 200, 200);
noTint();        // Disable tint()
```

Example 2

w02_code_02

```
1  PImage image0;
2
3  void setup() {
4
5    size(400, 400);
6    image0 = loadImage("Haeckel01.jpg"); // 450x450
7
8    image(image0, 0, 0, 200, 200);
9
10   tint(255,0,0);  // RED
11   image(image0, 200, 0, 200, 200);
12
13   tint(0,255,0);  // GREEN
14   image(image0, 0, 200, 200, 200);
15
16   tint(0,0,255);  // BLUE
17   image(image0, 200, 200, 200, 200);
18
19   tint(255,255,0,150);
20   image(image0, 50, 50, 300, 300);
21
22 }
```

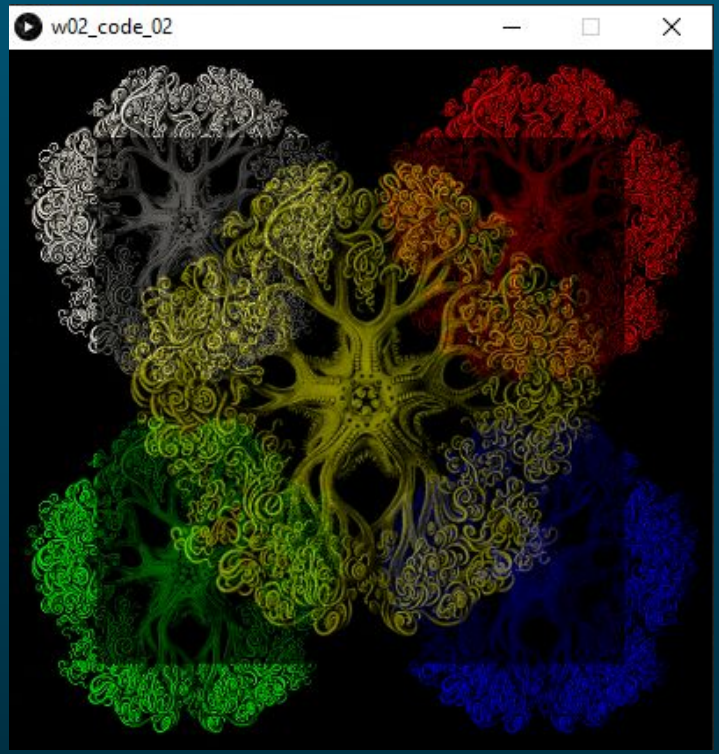


Image processing with filter()

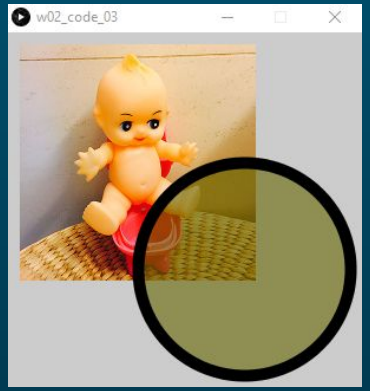
```
filter(<FILTER_NAME>);
filter(<FILTER_NAME>, <parameter>);
```

filters everything on the canvas using the specified filter.

Example

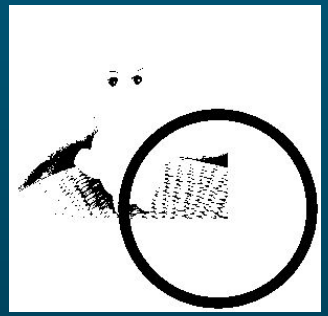
```
image(image0, 100, 100, 200, 200);
rect(0, 0, 200, 200);
filter(BLUR, 10); // Blurs everything
```

Image processing with `filter()`

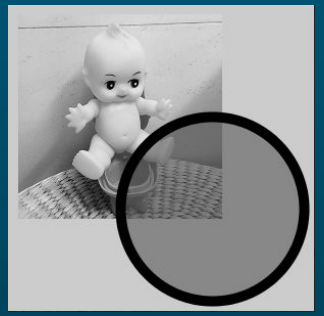


Original

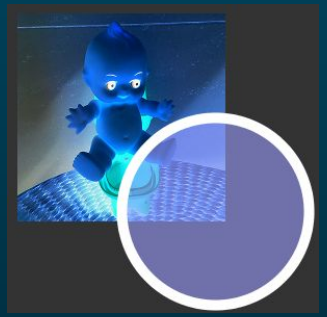
`filter(BLUR,10);`



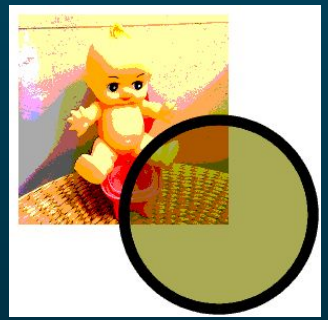
`filter(THRESHOLD);`



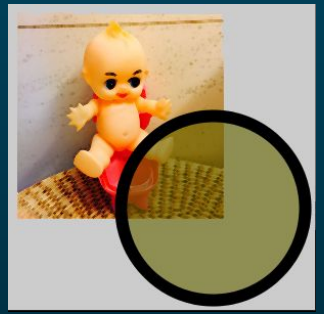
`filter(GRAY);`



`filter(INVERT);`



`filter(POSTERIZE,4);`

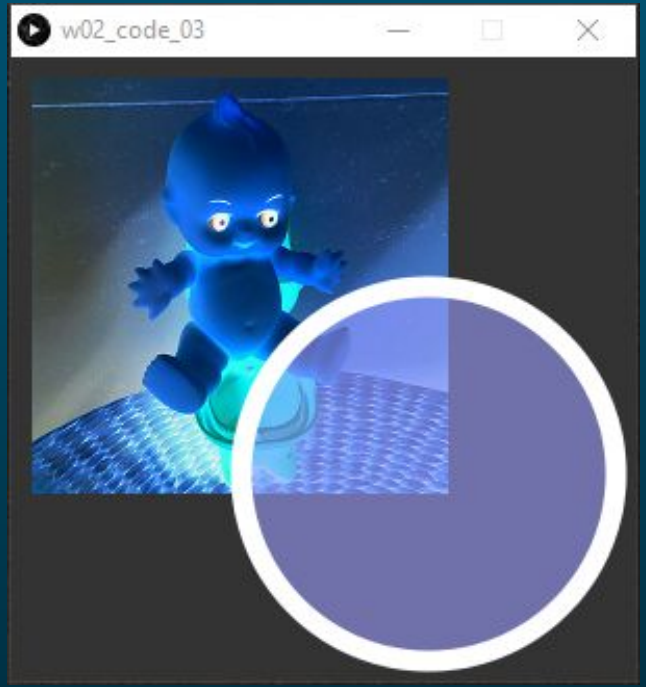


`filter(ERODE);`

Example 3

w02_code_03

```
PImage image0;

void setup() {

  size(300, 300);
  image0 = loadImage("baby.jpg"); // 450x450

  image(image0, 10, 10, 200, 200);
  fill(100,100,0,150);
  strokeWeight(10);
  ellipse(200,200,180,180);

//  filter(ERODE);          // Expand the DARK Areas
//  filter(DILATE);         // Expand the BRIGHT Areas
  filter(INVERT);        // Invert the colors
//  filter(GRAY);           // Turn into grayscale
//  filter(BLUR, 10);       // Blurs the image (filter radius = 10)
//  filter(POSTERIZE, 4); // Limit # of levels in each channel
//  filter(OPAQUE);         // Make whole display window OPAQUE
//  filter(THRESHOLD);      // To B/W based on a threshold (0.0 - 1.0)
}
```



More about PImage

For each `PImage` variable, there are additional properties and functions attached to it. Here are some:

Fields (Properties)

`.width` : width of the stored image

`.height` : height of the stored image

Example

```
int w = image0.width;
```

More about `PImage`

For each `PImage` variable, there are additional properties and functions attached to it. Here are some:

Methods (functions)

`.resize()` : resizes the stored image

`.get()` : returns the color of a pixel or
returns a sub-image

`.set()` : sets the color of a pixel or
replaces an area by another image

.resize() method of PImage

```
.resize(<new width>, <new height>);
```

w02_code_04

```
1  PImage image0;
2
3  void setup() {
4
5    size(400, 400);
6    image0 = loadImage("baby.jpg"); // 400x400
7    image(image0, 0, 0);
8
9    int w2 = image0.width / 2;
10   int h2 = image0.height / 2;
11   image0.resize(w2,h2);
12   tint(255,150);
13   image(image0, 0, 0);
14   image0.resize(w2/2,h2/2);
15   image(image0, 0, 0);
16
17 }
```

.get() method of PImage

`.get(<x>,<y>);`
returns the color of the pixel at `(x,y)` of the image.

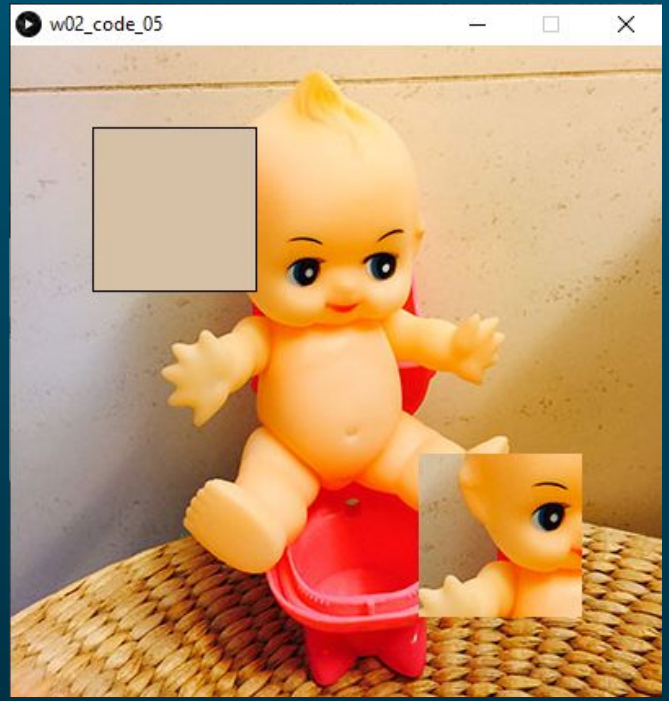
`.get(<x>,<y>,<w>,<h>);`
returns a `PImage` with its image content copied from the sub-region at `(x,y)` of size `w` x `h` of the image.

`.get();`
returns a `PImage` with its image content copied from the image.

Example 5

w02_code_05

```
1  PImage image0;
2
3  void setup() {
4
5    size(400, 400);
6    image0 = loadImage("baby.jpg"); // 400x400
7    image(image0, 0, 0);
8
9    color c = image0.get(100,100);
10   fill(c);
11   rect(50,50,100,100);
12
13   PImage subImage = image0.get(100,100,100,100);
14   image(subImage, 250, 250);
15
16 }
```



.set() method of PImage

`.set(<x>,<y>,<c>);`
replaces the color of the pixel at `(x,y)` with `'c'`.

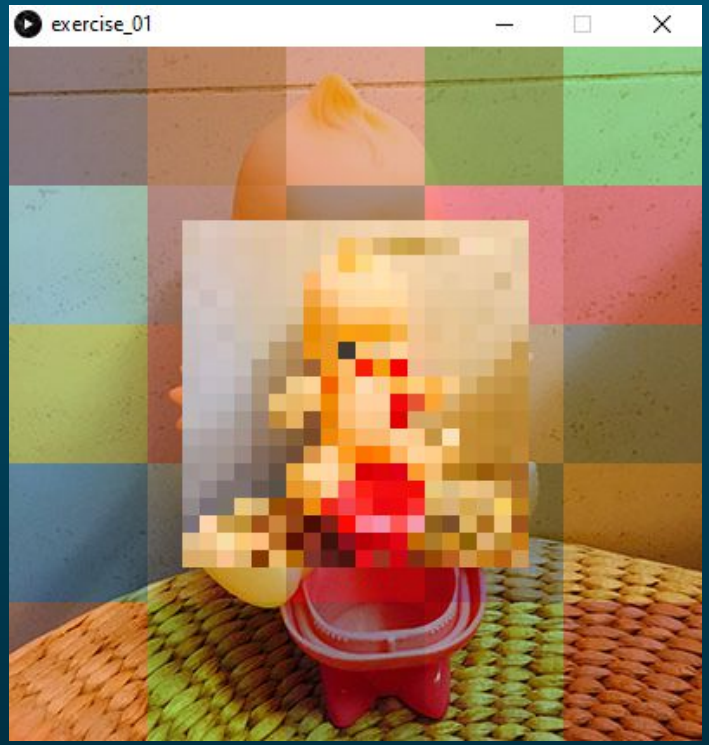
`.set(<x>,<y>,<src_image>);`
replaces an area at `(x,y)` with the image content copied from `'src_image'`.

Example 6

w02_code_06

```
PImage image0, image1;

void setup() {

  size(400, 400);
  image0 = loadImage("baby.jpg");  // 400x400
  image1 = loadImage("robot.jpg"); // 100x100
  color black = color(0);

  image0.set(200,100,black);
  image0.set(200,102,black);
  image0.set(202,100,black);
  image0.set(202,102,black);

  image0.set(200,200,image1);
  image(image0, 0, 0);

}
```

In-class Exercise



1. Create a block-by-block (You may redefine number of blocks > 3) tinted background, you will need the `.get()` method to obtain **sub-regions** of the image and display each block with **random tinting**. You may want to use `random(<range>)` to generate random tint colors.

2. Create a small pixelated version of the image and display it at the center.

`for-loop` example in Processing:

```
for (int i=0; i<10; i+=10){}
```