Week 03
Playing with Image Part 2

by Mike Wong. School of Creative Media

Quick Review of Part 1

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>)

```
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");
```

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
```

Changing image display with `tint()`

w02_code_02

```
PImage image0;

void setup() {

  size(400, 400);
  image0 = loadImage("Haeckel01.jpg"); // 450x450

  image(image0, 0, 0, 200, 200);

  tint(255,0,0);  // RED
  image(image0, 200, 0, 200, 200);

  tint(0,255,0);  // GREEN
  image(image0, 0, 200, 200, 200);

  tint(0,0,255);  // BLUE
  image(image0, 200, 200, 200, 200);

  tint(255,255,0,150);
  image(image0, 50, 50, 300, 300);

}
```

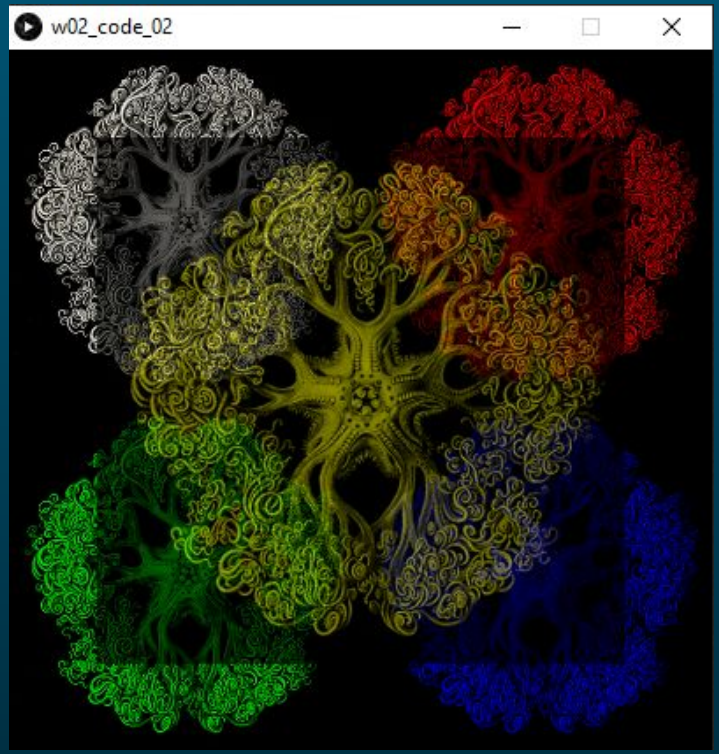
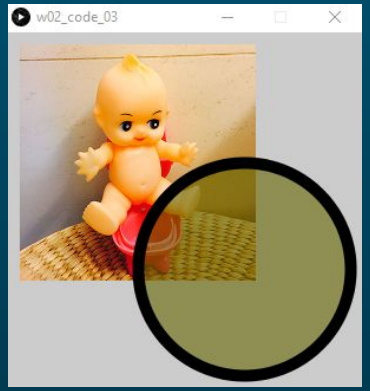
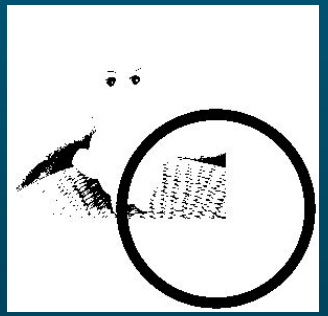


Image processing with `filter()`

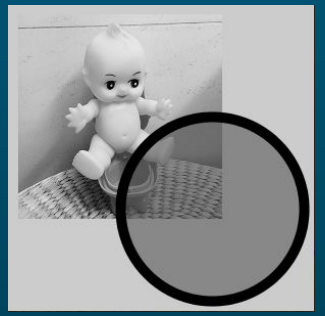


Original

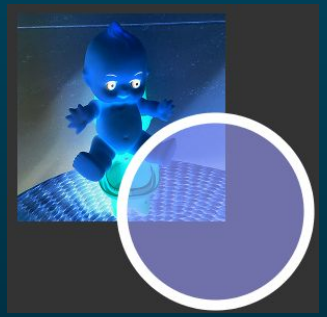
`filter(BLUR,10);`



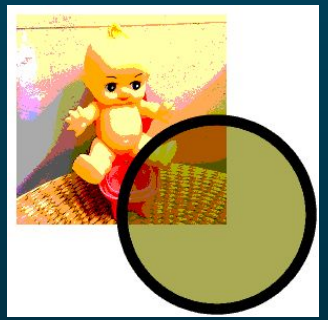
`filter(THRESHOLD);`



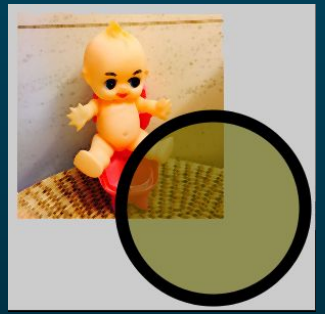
`filter(GRAY);`



`filter(INVERT);`



`filter(POSTERIZE,4);`



`filter(ERODE);`

Useful `PImage` fields

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;
```

Useful `PImage` methods

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

by Mike Wong. School of Creative Media

Playing with Image Part 2
More about `PImage`

Fields and Methods of PImage

Fields	Description
`.width`	image width
`.height`	Image height

Methods	Description
`.resize()`	resize the image
`.get()`	get color of a single pixel or a rectangle of pixels
`.set()`	set color of a single pixel or write an image into another

More Methods of PImage

Methods	Description
`.copy()`	copies the entire image
`.mask()`	masks the image using another image as alpha channel
`.filter()`	filters the image using the specified filter
`.blend()`	blends the image with another one using various blending modes (PhotoShop alike)
`.save()`	Saves the image to a `.tiff`, `.tga`, `.png` or `.jpg` file

.copy() method of PImage

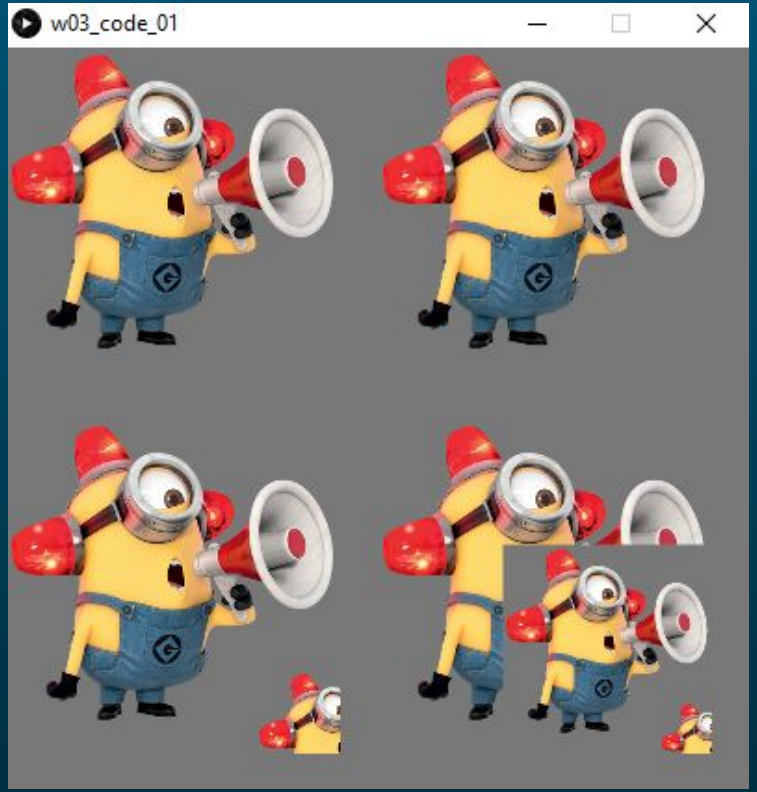
copies region within or across images.

```
.copy();  // returns a copy of the image

// self-copy a region to another region
.copy(sx,sy,sw,sh, dx,dy,dw,dh);

// copy a region from the another image 'src'
.copy(src, sx,sy,sw,sh, dx,dy,dw,dh);
```

Example 1 - Copy

w03_code_01

```
PImage original, clone;

void setup() {

  size(400, 400);
  background(120);

  //  Load the source image into 'original'
  original = loadImage("fg.png");  // size: 400x400
  image(original, 0, 0, 180, 180); // Display: Upper Left

  //  Simple .copy() from 'original' to 'clone'
  clone = original.copy();
  image(clone,  200, 0, 180, 180);  //  Upper Right

  //  SELF-COPY within the 'clone'
  //  from (0,0) of size 200x200
  //  to (300,300) of size 100x100
  clone.copy(0,0,200,200, 300,300,100,100);
  image(clone, 0, 200, 180, 180);   //  Lower Left

  //  COPY from 'clone' to 'original'
  //  from (0,0) of size 400x400
  //  to (150,150) of size 250x250
  original.copy(clone, 0,0,400,400, 150,150,250,250);
  image(original, 200, 200, 180, 180);  //  Lower Right

}
```



`.mask()` method of `PImage`

Uses another image's **`BLUE`** channel as ALPHA.

```
.mask(<maskImage>);
```

Example 2 - Simple Masking

w03_code_02

```
1  PImage wallImage, maskImage;
2
3  void setup() {
4
5    size(400, 400);
6    background(120);
7
8    wallImage = loadImage("wall.jpg");   // 400x400
9    maskImage = loadImage("mask.jpg");   // 400x400
10
11   image(wallImage,   0,0, 200,200);  // Upper Left
12   image(maskImage, 200,0, 200,200);  // Upper Right
13
14   wallImage.mask(maskImage);
15   image(wallImage, 100,200, 200,200);  // Upper Right
16
17 }
```

Example 3 - Sliding Mask

w03_code_03

```
PImage wallOriginal, maskOriginal;
PImage slidingImage, slidingMask;

void setup() {
  size(400, 400);
  wallOriginal = loadImage("wall.jpg");  // 200x200
  maskOriginal = loadImage("mask.jpg");  // 200x200
  slidingMask = maskOriginal.copy();     // Makes it 200x200
  noFill();
}

void draw() {
  background(120);
  image(maskOriginal, 0,0);

  //  Variable 'os' cycles from 0 -> 199 continuously
  int os = frameCount % 200;
  int widthL = os;        // LEFT Hand Region  (0,0,widthL,200)
  int widthR = 200 - os;  // RIGHT Hand Region (os,0,widthR,200)

  //  Copies LEFT Region
  slidingMask.copy(maskOriginal,  0,0, widthL,200, 200-os,0, widthL,200);
  //  Copies RIGHT Region
  slidingMask.copy(maskOriginal, os,0, widthR,200, 0,0, widthR,200);

  //  Apply the moving mask
  slidingImage = wallOriginal.copy();
  slidingImage.mask(slidingMask);

  //  Display
  image(slidingMask,   0,200);
  image(slidingImage, 200,100);

  //  Visualization of RIGHT Hand Region Copying
  stroke(0,255,0);
  rect( os,0,   widthR,200);
  rect(  0,200, widthR,200);
}
```

.filter() method of PImage

filters the image with the given filter.

```
.filter(<FILTER_NAME>, [param]);
```

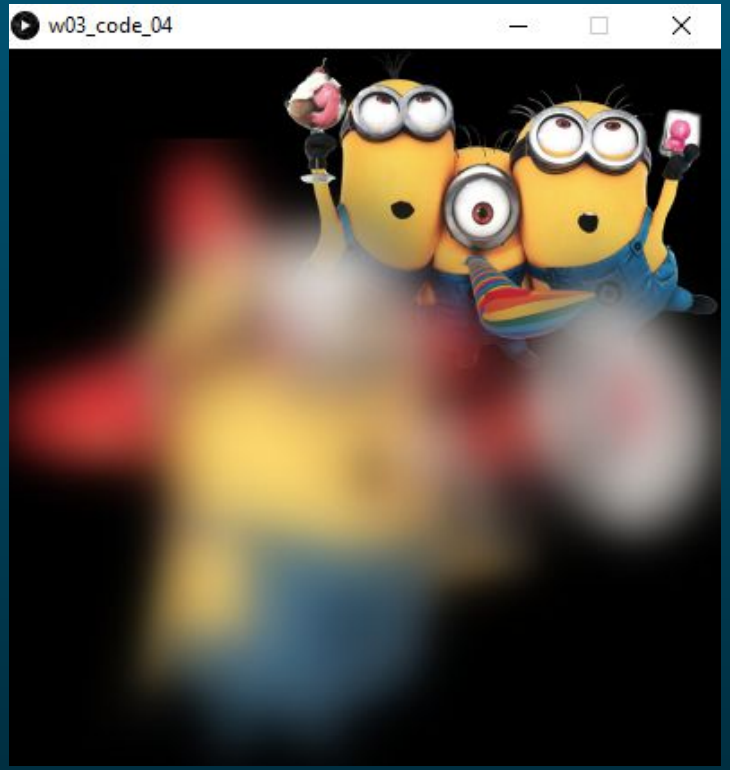
Supports the following standard filters:

THRESHOLD, GRAY, OPAQUE, INVERT,
POSTERIZE, BLUR, ERODE, DILATE

Example 4 - Simple DOF

w03_code_04

```
PImage fg, bg, fgDOF, bgDOF;
float  maxBlur = 15;

void setup() {
  size(400, 400);
  fg = loadImage("fg.png");  // 400x400
  bg = loadImage("bg.png");  // 400x400
}

void draw() {
  background(0);
  //  maps 'mouseX' to the focus point
  float focus = map(mouseX,0,width,0,maxBlur);

  //  Obtain fresh copies of fg & bg
  bgDOF = bg.copy();
  fgDOF = fg.copy();
  bgDOF.filter(BLUR, focus);
  fgDOF.filter(BLUR, maxBlur - focus);

  //  Display
  image(bgDOF, 150, 0, 250, 250);
  image(fgDOF, 0, 50);
}
```



.blend() method of PImage

`.blend(sx,sy,sw,sh, dx,dy,dw,dh, <mode>);`
self-blends the ‘source’ region with the ‘destination’ region using the given blend mode.

`.blend(src, sx,sy,sw,sh, dx,dy,dw,dh, <mode>);`
blends the ‘source’ region from a source image with the ‘destination’ region using the given blend mode.

Supported blend modes: `BLEND, ADD, SUBTRACT, DARKEST, LIGHTEST, DIFFERENCE, EXCLUSION` …

Example 5 - Blending

w03_code_05

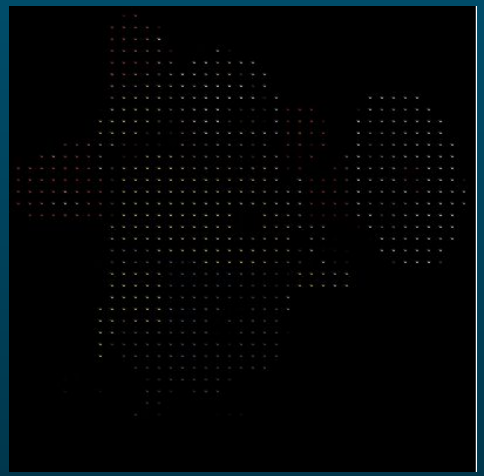
```
PImage A, B, C;

void setup() {
  size(600, 400);
  A = loadImage("fg.png");    // 400x400
  B = loadImage("forest.jpg");  // 400x400
  background(0);

  C = B.copy();
  C.blend(A, 0,0, 400,400, 0,0, 400,400, BLEND);
  image(C, 0, 0, 180, 180);

  C = B.copy();
  C.blend(A, 0,0, 400,400, 0,0, 400,400, ADD);
  image(C, 200, 0, 180, 180);

  C = B.copy();
  C.blend(A, 0,0, 400,400, 0,0, 400,400, SUBTRACT);
  image(C, 400, 0, 180, 180);

  C = B.copy();
  C.blend(A, 0,0, 400,400, 0,0, 400,400, SCREEN);
  image(C, 0, 200, 180, 180);

  C = B.copy();
  C.blend(A, 0,0, 400,400, 0,0, 400,400, OVERLAY);
  image(C, 200, 200, 180, 180);

  C = B.copy();
  C.blend(A, 0,0, 400,400, 0,0, 400,400, DIFFERENCE);
  image(C, 400, 200, 180, 180);
}
```

Example 6 - Displacement

w03_code_06

```
PImage src, dst;

void setup() {
  size(600, 300);
  src = loadImage("fg.png");    // 400x400
  //  To create an empty PImage, we may use createImage()
  dst = createImage(src.width, src.height, 0);

  //  Go through our 'dst' image pixel-by-pixel
  for (int y = 0; y < dst.height; y++) {
    for (int x = 0; x < dst.width; x++) {
      int rx = int(random(-30,30)); // get an integer from -30 - 30
      int ry = int(random(-30,30)); // get an integer from -30 - 30
      int sx = x + rx;
      int sy = y + ry;
      sx = safe(sx, 0, src.width);
      sy = safe(sy, 0, src.height);
      color srcColor = src.get(sx,sy);
      dst.set(x,y,srcColor);
    }
  }
  background(0);
  image(src,   0, 0);
  image(dst, 300, 0);

  dst.save("displacement.jpg");  //  save only dst
  save("canvas.jpg");            //  save whole canvas
}

//  Make sure the coordinates are valid range
int safe(int v, int min, int max) {
  if (v < min) return max + v;
  else if (v >= max) return v - max;
  return v;
}
```

In-class Exercise



1. Load the given '`fg.png`' as your source image. Sample the color of pixels sparsely, such as sample only one pixel for every 10 pixels (similar to what you did for pixelation in Week 02).

2. For each pixel you have sampled, replace it with a 'star' image using the given '`star.jpg`' (this image has no alpha channel).

3. When you move your mouse from LEFT to RIGHT, the size of each star should increase, and they should also move away from their original position randomly.