



# Code ↔ Nature

Processing @ UdK Raum 115

Part 3

April - June, 2016

# Load and display images

The same way float is a type that can contain numbers, PImage is a type that can contain bitmap images, and PShape can contain vector images.

## Loading JPG, PNG, GIF, TIF, TGA

```
// Declare a PImage variable
PImage catPhoto;

// Load an image into the variable
catPhoto = loadImage("cat.png");

// Display the image
image(catPhoto, 100, 100);
```

## Loading SVG

```
// Declare a PShape variable
PShape logo;

// Load a vector image into the variable
logo = loadShape("logo.svg");

// Display the vector image
shape(logo, 200, 200);
```

# Load and display images

To add an image to your Processing sketch you have two options:

1) drag and drop the image on top of your sketch window

or

2) use the "sketch/Add file..." menu option

A "data" folder will be created inside your sketch folder, and the file will be placed inside.

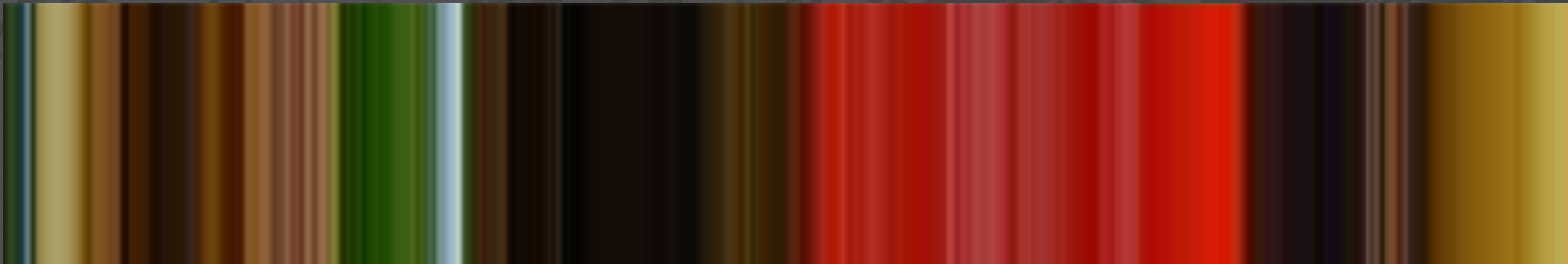
Right:	<code>loadShape("myfile.svg");</code>
Wrong:	<code>loadShape("data/myfile.svg");</code>

Exercise: write a program that loads an image once, then uses a for loop to display it 5 times on each animation frame.



# Read image pixel color using .get()

```
// Use colors coming from an image to set the stroke color
PImage img;
size(600, 200);
// load a random image from a website
img = loadImage("http://lorempixel.com/600/600/#.png");
// cover the screen with vertical lines
for (int x=0; x<width; x=x+1) {
  // read the color of a pixel from the image
  color c = img.get(x, 300);
  stroke(c);
  line(x, 0, x, height);
}
```



# Read image pixel color using .get()

```
// Use colors coming from an image to control the sizes of circles
PImage img;
size(1200, 200);
background(#4c4c4c);
img = loadImage("http://lorempixel.com/1200/200/#.png");
for (int x=50; x<width-50; x=x+1) {
    color c = img.get(x, 100);
    float sz = brightness(c) / 4;
    ellipse(x, 100, sz, sz);
}
save("result.png");
```



# Rotate, translate, scale

Functions that modify the origin and the axes:

## **rotate(angle)**

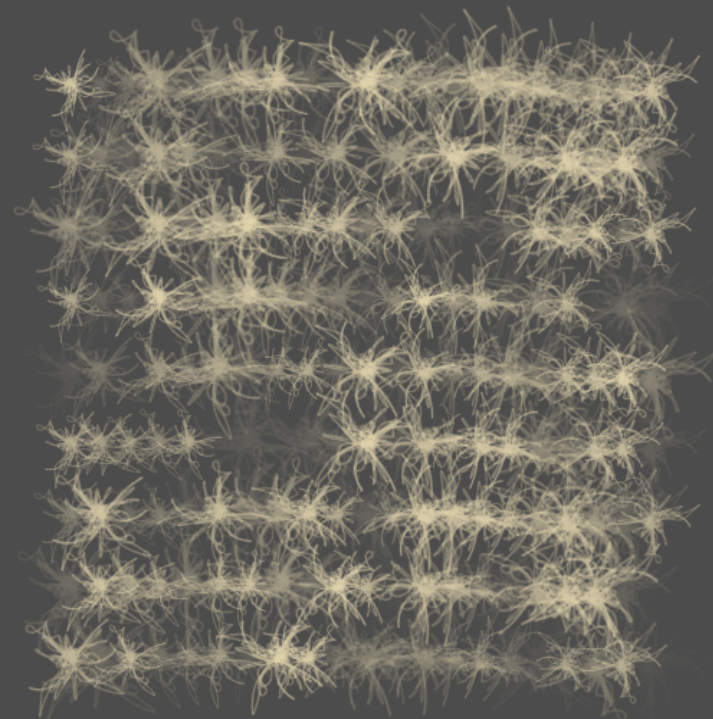
Used to rotate objects.  
Rotation always takes place  
around the origin.

## **translate(x, y)**

Moves the origin to a  
new position.

## **scale(k)**

Used to enlarge or shrink objects  
drawn after calling this function.  
The scaling center is the origin.



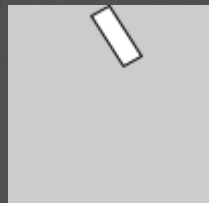


# Rotate, translate, scale

A sketch to visualize the effect of translate, rotate and scale:

<http://www.openprocessing.org/sketch/196409>

The order of the operations produces different results:



```
translate(50, 0);  
rotate(1);  
rect(0, 0, 30, 10);
```



```
rotate(1);  
translate(50, 0);  
rect(0, 0, 30, 10);
```

A common pattern is to first call translate to position the axes, then call rotate or scale (or both).

# Rotate an existing rectangle

```
void setup() {  
  size(600, 400);  
  rectMode(CENTER);  
}  
void draw() {  
  background(222);  
  
  // axis aligned rectangle  
  rect(400, 100, 60, 20);  
  
  // rotated rectangle  
  translate(400, 100);  
  rotate(1);  
  rect(0, 0, 60, 20);  
}
```



Notice how the second rectangle is drawn at (0, 0).

Since rotations always happen around the origin, if we want to rotate a rectangle around its center we need to draw it at (0, 0).

We use translate to place the rectangle on the desired location.



# Rotating rectangle animation

```
void draw() {  
    background(255);  
  
    translate(30, 50);  
    rotate(frameCount * 0.05);  
    rect(0, 0, 30, 30);  
}
```

We always draw the rectangle on the same place, with the same size.

Try drawing a second rectangle next to the first one and observe the result.

Tip: `rectMode(CENTER);` makes `rect()` behave like `ellipse()`: the two first arguments specify the center of the rectangle instead of the top left corner.

# Rotating rectangle animation

What if we want to draw two rotating rectangles?

```
void setup() {  
    rectMode(CENTER);  
}  
void draw() {  
    background(255);  
  
    translate(30, 50);  
    rotate(frameCount * 0.05);  
    rect(0, 0, 30, 30);  
  
    translate(70, 50);  
    rotate(frameCount * 0.05);  
    rect(0, 0, 30, 30);  
}
```

# Rotating rectangles animation

What if we want to draw two rotating rectangles?

The effect of translate, rotate and scale are cumulative:

```
translate(20, 5);  
translate(20, 5);
```

has the same effect as:

```
translate(40, 10);
```

The cumulative effect only takes place until draw() is called again. Then the axes reset automatically.

```
void setup() {  
  rectMode(CENTER);  
}  
void draw() {  
  background(255);  
  
  translate(30, 50);  
  rotate(frameCount * 0.05);  
  rect(0, 0, 30, 30);  
  
  translate(70, 50);  
  rotate(frameCount * 0.05);  
  rect(0, 0, 30, 30);  
}
```



# Rotating rectangles animation

What if we want to draw two rotating rectangles?

We can use `pushMatrix()` and `popMatrix()` to avoid the cumulative effect of the transformations.

`pushMatrix()` and `popMatrix()` must always be used in pairs.

`pushMatrix()` saves the state of the coordinate system().

`popMatrix()` restores the state previously saved.

```
void setup() {  
  rectMode(CENTER);  
}  
void draw() {  
  background(255);  
  
  pushMatrix();  
  translate(30, 50);  
  rotate(frameCount * 0.05);  
  rect(0, 0, 30, 30);  
  popMatrix();  
  
  pushMatrix();  
  translate(70, 50);  
  rotate(frameCount * 0.05);  
  rect(0, 0, 30, 30);  
  popMatrix();  
}
```

# Using typography

1. This line of code will tell you the names of the fonts found on your system:

```
printArray(PFont.list());
```

2. Use the output of the previous command to choose one of the font names. Add the font to your program like this (32 is the desired default text size):

```
PFont myFont1 = createFont("Arial", 32);
```

3. Set the font you just created as the active font. Following calls to text() will use that font.

```
textFont(myFont1);
```

4. Write something on the screen:

```
text("watermelon", 20, 100);
```

Set the text color calling fill(). The text size can be set like this: textSize(48).

# Code ↔ Nature

April - June 2016 // UdK - Berlin

Abe Pazos

@fun\_pro | fun@funprogramming.org

<http://hamoid.com> | <http://funprogramming.org>