

# **Introduction to Processing**

The Basics(Python Version)

# Processing

- Processing started by Ben Fry and Casey Reas while both were graduate students at MIT Media Lab in 2001.
- The original language for Processing is Java. **We will use the Python version.**
- Designed for visual artists with limited programming experience who want to create art without knowing complicated Java syntax.
- In its current version, hundred of libraries have been written for computer vision, data visualization, music composition, networking, 3D drawings and programming electronics.

# Processing

Processing was created originally for the Java language. For this reason, the interface to Processing's Python version is not very "Pythonic".

I wrote some code to hide some of this interface and make it flow better with Python. Download the zip file that contains this code on our course website [here](#).

Once you unzip the contents and open it with Processing. There should be three files:

processing\_py.pyde(DO NOT MODIFY THIS FILE)

game.py(write all of your code here)

There is also a **data** folder where you should put all of your images for your game.

# game.py

All of your code should go here in game.py.

You will need to implement(provide code for) **three methods/functions**:

- 1) **def \_\_init\_\_(self)**: Declare and initialize all your game/application variables.
- 2) **def on\_draw(self)**: Called automatically 60 times a second to draw objects. Write code to draw all objects here.
- 3) **def on\_update(self)**: Called automatically 60 times a second to update our objects. Write code to update all objects here(for animation).

# Sketch

First declare and initialize all variables in `__init__`

```
class Window:
```

```
    def __init__(self):
```

```
        """ Initialize all variables here. """
```


**`__init__` only runs ONCE.**



```
    def on_draw(self):
```

```
        """ Called automatically 60 times a second to draw all objects. """
```


**`on_draw` runs automatically 60 times a second  
to draw all images**



```
    def on_update(self):
```

```
        """ Called automatically 60 times a second to update all objects. """
```

**`on_update` runs automatically 60 times a second  
to update variables**



# Creating Variables

```
class Window:
```

**When declaring/initializing a global variable that is used throughout the game, use self and the dot notation.**

```
def __init__(self):  
    """ Initialize all variables here. """  
    self.x = 10  
    y = 5
```

**The y variable here does not have the “self.” prefix. Consequently, it only exists locally here in init.**

```
def on_draw(self):  
    """ Called automatically 60 times a second to draw all objects."""  
    print(self.x)  # valid!  
    print(y)       # error! y does not exist here!
```

```
def on_update(self):  
    """ Called automatically 60 times a second to update all objects."""  
    self.x += 5    # valid!  
    y += 1         # error! y does not exist here!
```

# Updating Variables

What values are printed on the console in the following program?

```
class Window:
```

```
    def __init__(self):  
        """ Initialize all variables here. """  
        self.x = 10
```

```
    def on_draw(self):  
        """ Called automatically 60 times a second to draw all objects."""  
        print(self.x)
```

```
    def on_update(self):  
        """ Called automatically 60 times a second to update all objects."""  
        self.x += 5
```

Answer: self.x has value:  
10 in the first frame  
15 in the second frame  
20 in the third frame  
etc...

# Animation

class Window:

Animation only takes five lines of code!

```
def __init__(self):
```

```
    """ Initialize all variables here. """
```

```
    self.x = WIDTH/2
```

```
    self.y = HEIGHT/2
```

```
def on_draw(self):
```

```
    """ Called automatically 60 times a second to draw all objects."""
```

```
    # fill(red, green, blue)
```

```
    fill(255, 0, 0)
```

draw red circle at (self.x, self.y)

```
    ellipse(self.x, self.y, 300, 300)
```

diameter = 300 pixels

```
def on_update(self):
```

```
    """ Called automatically 60 times a second to update all objects."""
```

```
    self.x += 5
```

move circle 5 pixels to the right

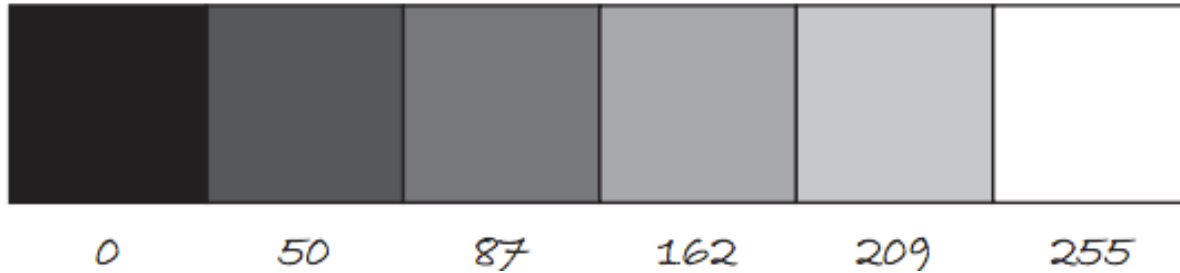
Repeat 60 times a second!



# Color

Color is defined by a range of numbers.

In grayscale, 0 is black, 255 is white and any color in between is a shade of gray ranging from black to white.



# Color

RGB Color is determined by three parameters. Each parameter is in the range 0-255. The first indicates the degree of red(R), the second the degree of green(G) and the last the degree of blue(B).

- Red + green = yellow
- Red + blue = purple
- Green + blue = cyan (blue-green)
- Red + green + blue = white
- No colors = black

# Some Methods for Drawing Shapes

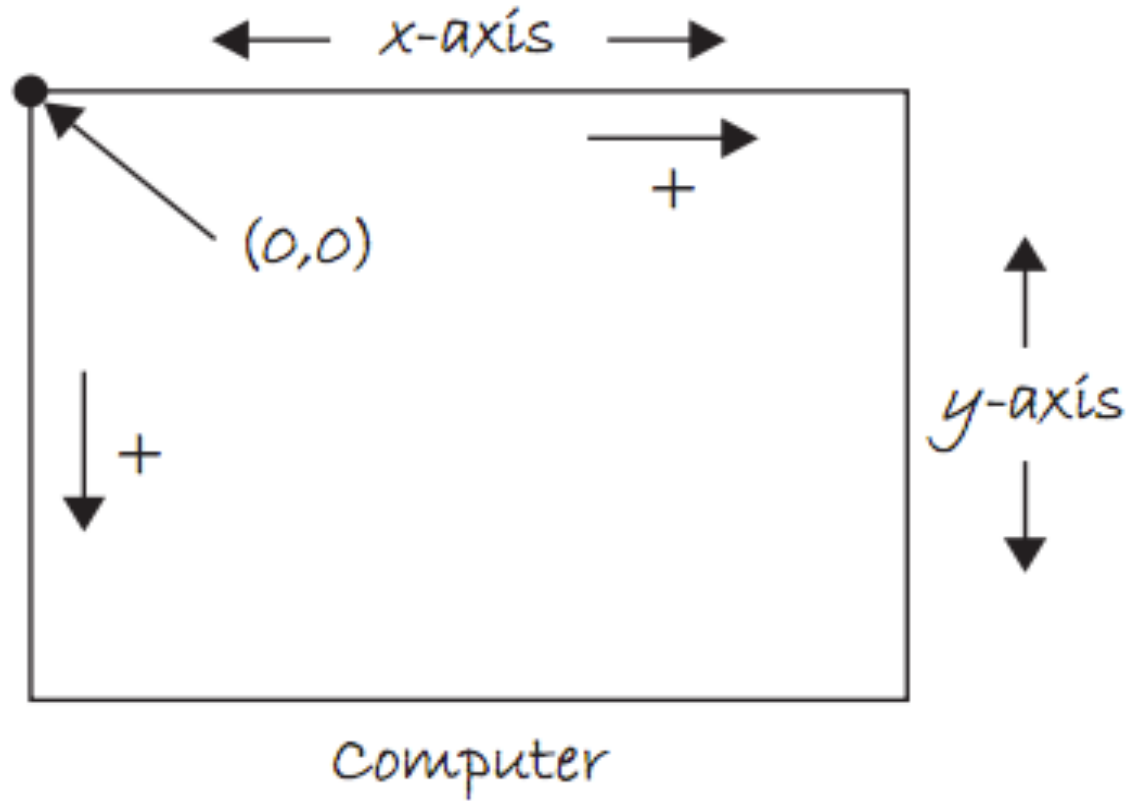
**fill(r, g, b)** : By calling fill BEFORE a shape will set the color of the shape. Call it again before drawing another shape to change color.

**line(x1, y1, x2, y2)** : draw line through (x1,y1) and (x2,y2).

**ellipse(x, y, width, height)** : center of ellipse is (x,y); width and height are the lengths of the axes.

**rect(x, y, width, height)** : center of the rectangle is (x,y)

# The Coordinate System



# Adding Text

The `text(str, x, y)` function draws text on the screen. You can set the text size and color by using `textSize(s)` and `fill(r, g, b)` before drawing the text.

```
textSize(32);  
fill(255, 0, 0);  
text("Hello, World!", 100, 200);
```

# The Console

Messages can be printed on the console(for error-checking purposes, etc..) by using the command `print()` .

```
print(4);  
print(4 + 3/2);  
print("Hello, world");
```

# Download Processing

Download Processing!

<http://www.processing.org>