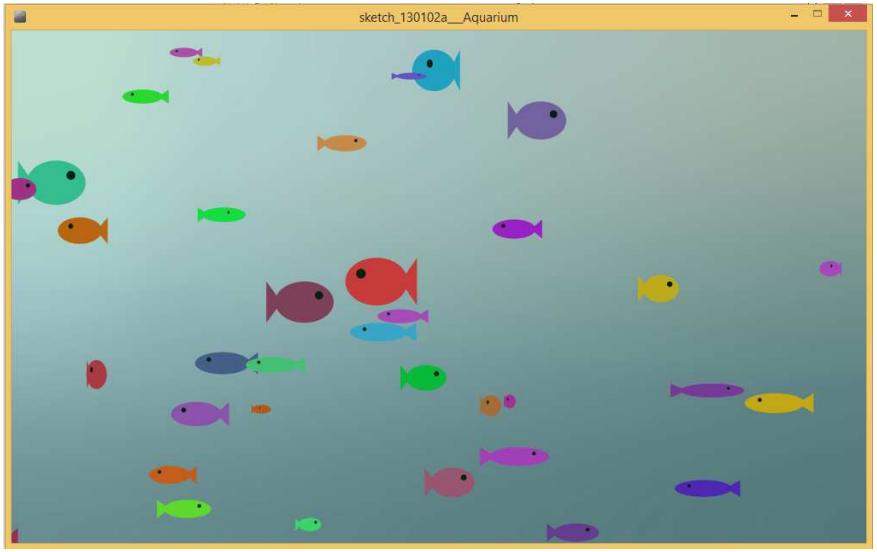
Visual applications with Processing and Java

What is Processing

Processing is a programming language based on Java, tailored for visual applications

The objective is to quickly sketch some idea an produce something that other people can understand

- i.e. **rapid prototyping** of something that an user can **see** and **interact** with
- *An image is worth a thousand words*



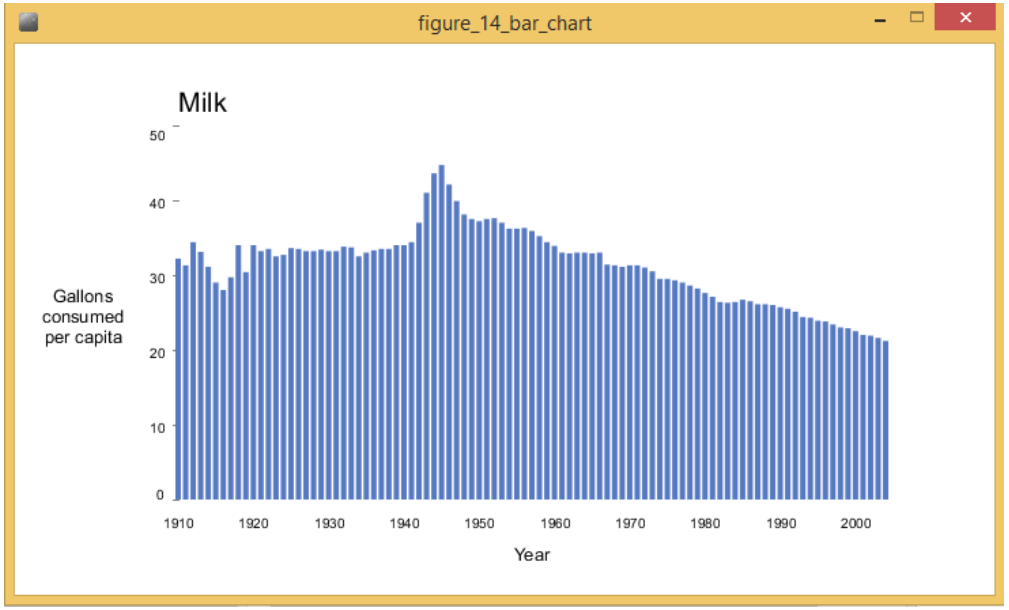
Drawing pretty stuff

What is Processing

Processing is a programming language based on Java, tailored for visual applications

The objective is to quickly sketch some idea an produce something that other people can understand

- i.e. **rapid prototyping** of something that an user can **see** and **interact** with
- *An image is worth a thousand words*



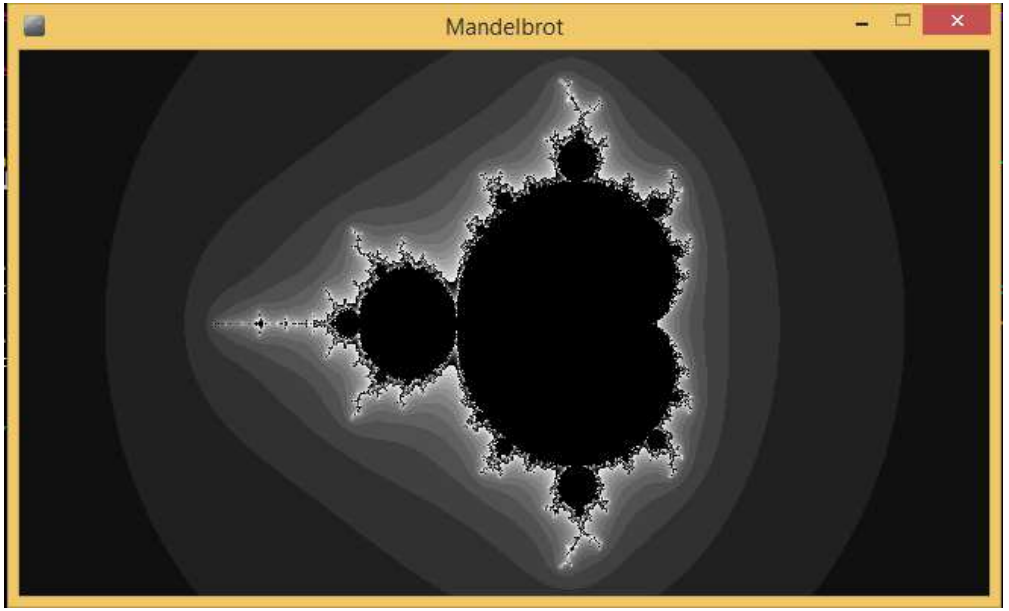
Visualizing data

What is Processing

Processing is a programming language based on Java, tailored for visual applications

The objective is to quickly sketch some idea an produce something that other people can understand

- i.e. **rapid prototyping** of something that an user can **see** and **interact** with
- *An image is worth a thousand words*



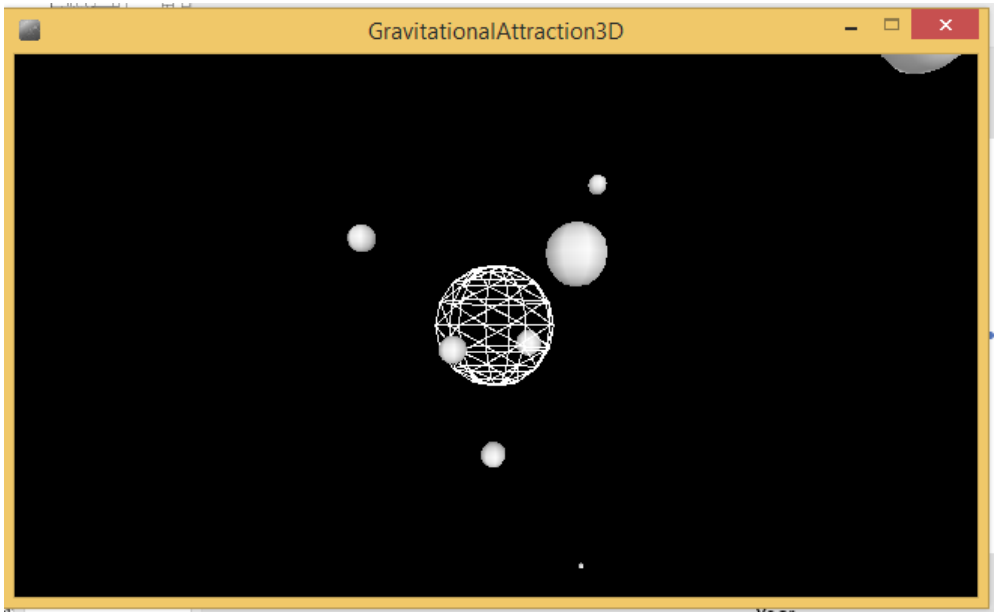
Plotting mathematical functions

What is Processing

Processing is a programming language based on Java, tailored for visual applications

The objective is to quickly sketch some idea an produce something that other people can understand

- i.e. **rapid prototyping** of something that an user can **see** and **interact** with
- *An image is worth a thousand words*



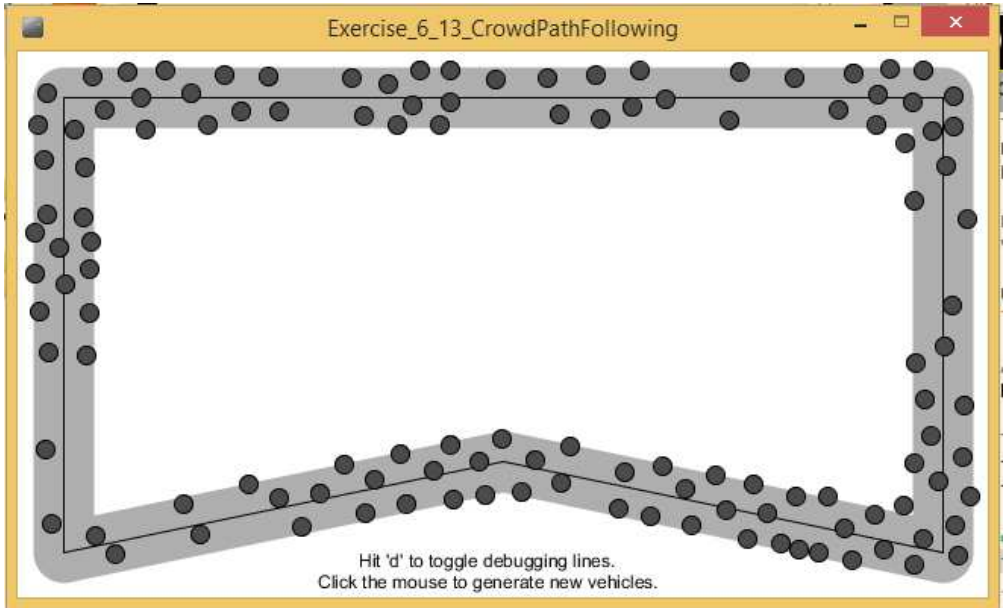
Simulating Newton's gravitation laws

What is Processing

Processing is a programming language based on Java, tailored for visual applications

The objective is to quickly sketch some idea an produce something that other people can understand

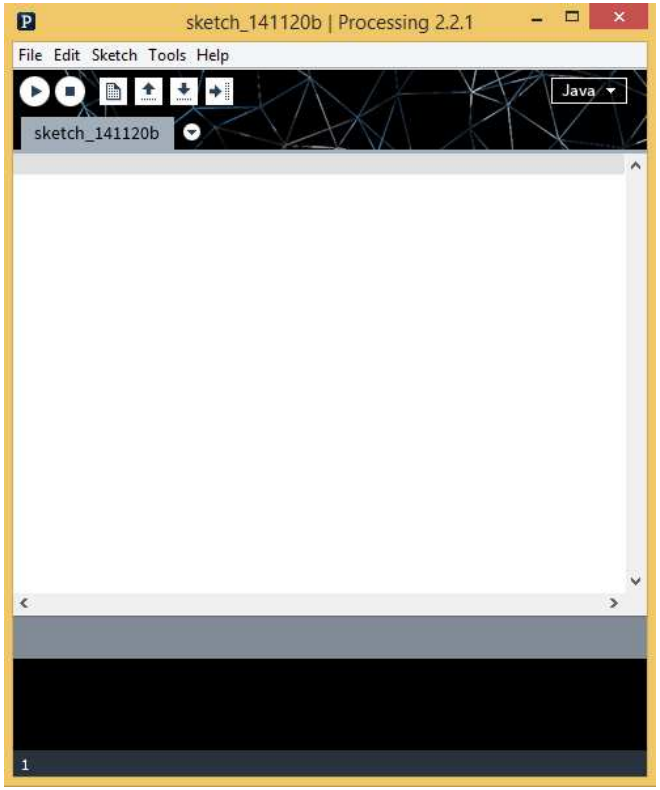
- i.e. **rapid prototyping** of something that an user can **see** and **interact** with
- *An image is worth a thousand words*



Simulating Artificial Intelligence (Little robots following a path while avoiding each other)

The Processing Interactive Development Environment (IDE)

Go get it at https://processing.org/download/



We will review most of the COMP 202 Java concepts within this Processing window

Processing

Processing is a programming language based on Java

It facilitates building graphical user interfaces (GUI), by abstracting some of the lower level concepts from Java

- You don't have to deal with the Java way of drawing things on the screen (e.g. see code for plotting in A3, or the Country map in A4)
- It *hides* from the programmer some of the basic constructs from Java
- **But you can still use Java code in processing**
  - Using Java data types (int, double, String)
  - Importing standard Java libraries (ArrayList, Hashtable, Math, Random, etc)
  - Creating your own object classes
  - Reading and writing files
  - Handling Exceptions

Moving on from our console (text input and output) applications

Hello world in Java

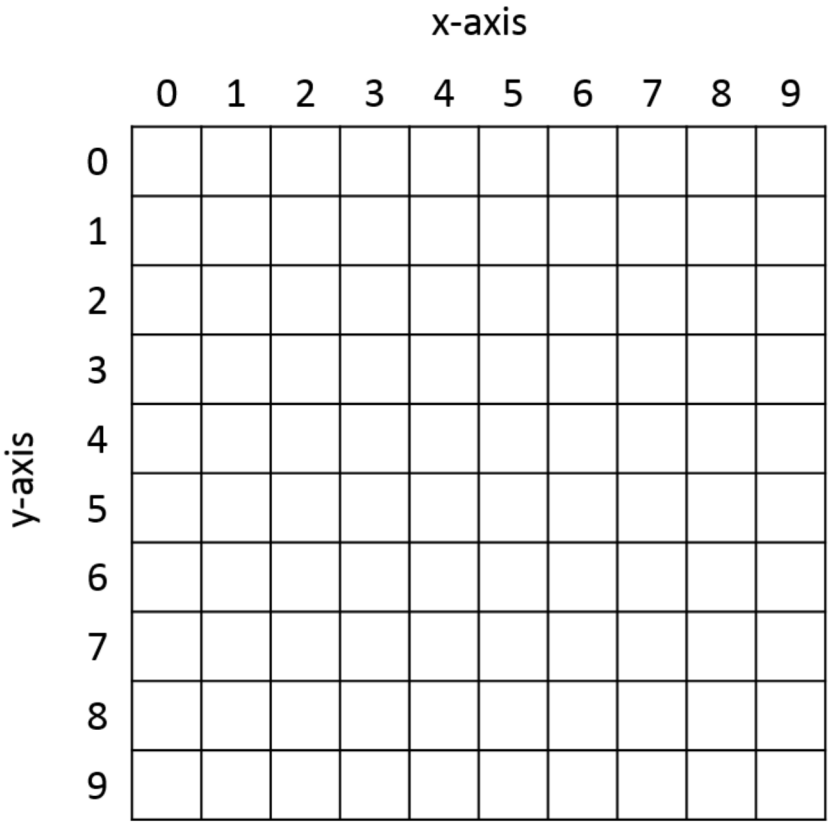
```
    public class HelloWorld{
        public static void main(){
            System.out.println("Hello World"); // printing in console
        }
    }

```

Hello world in Processing

```
    println("Hello World"); // printing in console

    size(640, 480); // setting up a window that is 640 pixels wide, by 320 pixels tall
    text("Hello World", 320, 240); // print hello world in the middle of the window

```

Pixel coordinate system

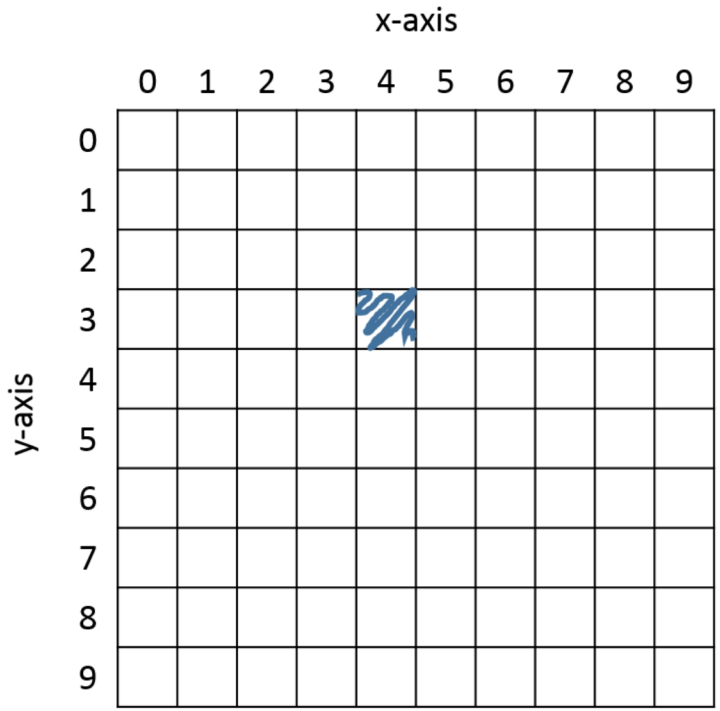
Going back to Assignment 2




```
1    // setting the size of our "canvas"
2    size(10, 10);
3
```

Pixel coordinate system

Going back to Assignment 2



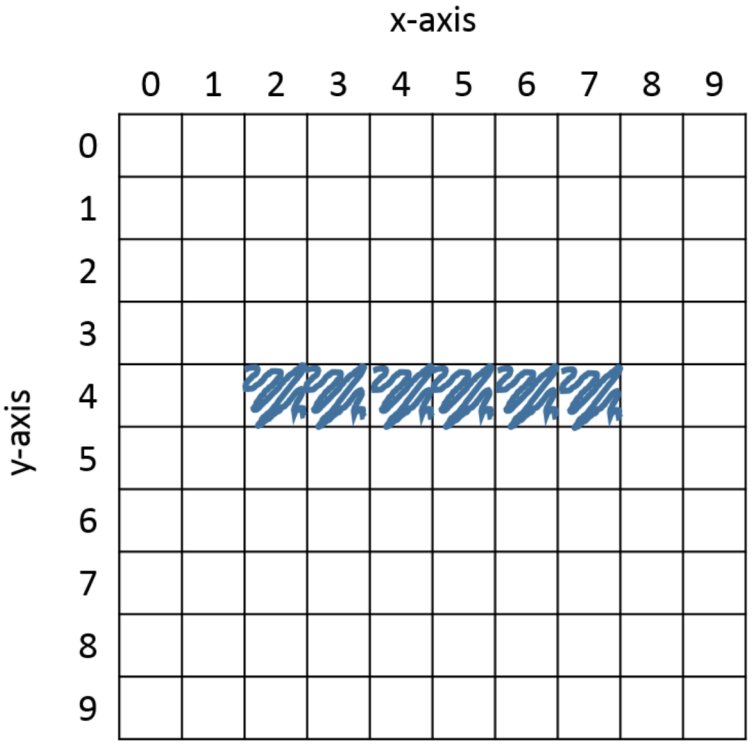
```
1    // setting the size of our canvas
2    size(10, 10);
3
4    // drawing a single point
5    point(4,3);
6
```

`point(x,y)` draws a point centered at pixel coordinates (x,y)

`strokeWeight(pixels)` lets you set the size of the point

Shapes

Drawing a line

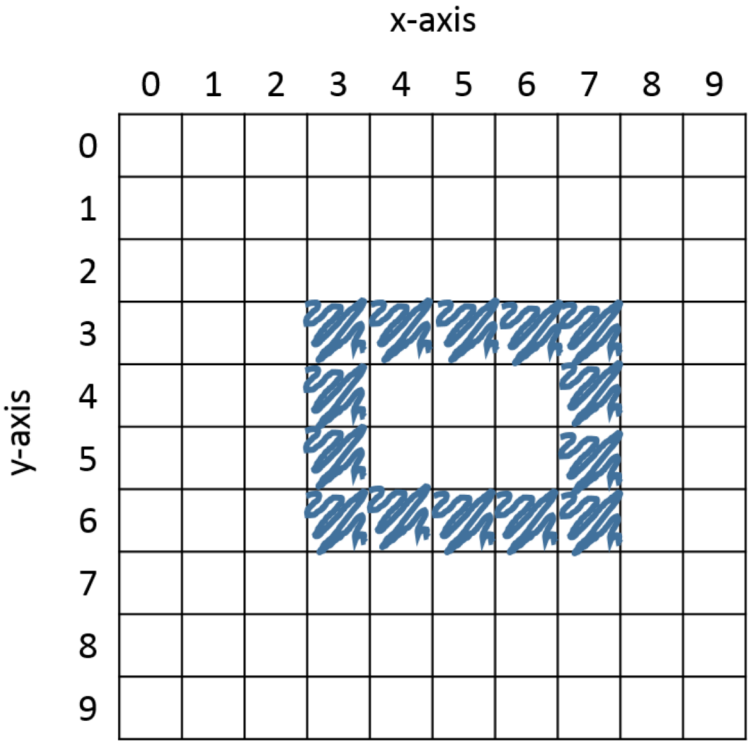


```
1    // setting the size of our canvas
2    size(10, 10);
3
4    // drawing a line
5    line(2,4,7,4);
6
```

`line(x1,y1,x2,y2)` draws a line segment from (x1,y1) to (x2,y2)

Shapes

Drawing a rectangle

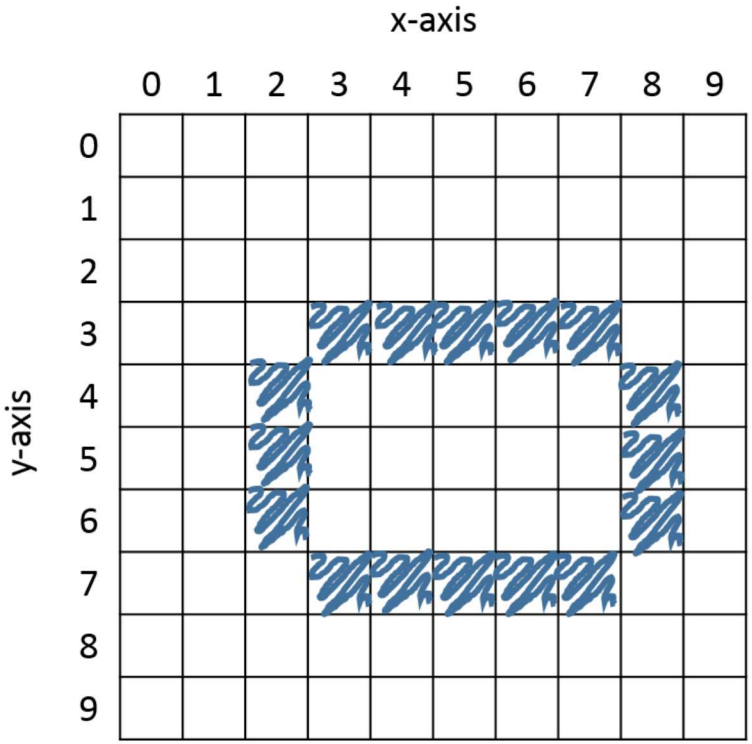


```
1    // setting the size of our canvas
2    size(10, 10);
3
4    // drawing a rectangle
5    rect(3,3,5,4);
6
```

`rect(x,y,w,h)` draws a rectangle whose top left corner is at (x,y) and its size is w by h pixels

Shapes

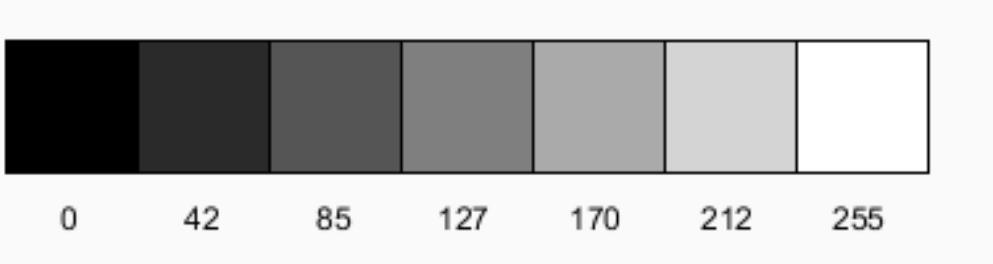
Drawing an ellipse



```
1    // setting the size of our canvas
2    size(10, 10);
3
4    // drawing an ellipse
5    ellipse(5,5,7,5);
6
```

`ellipse(x,y,w,h)` draws an ellipse centered at (x,y) with a size of w by h pixels

Color



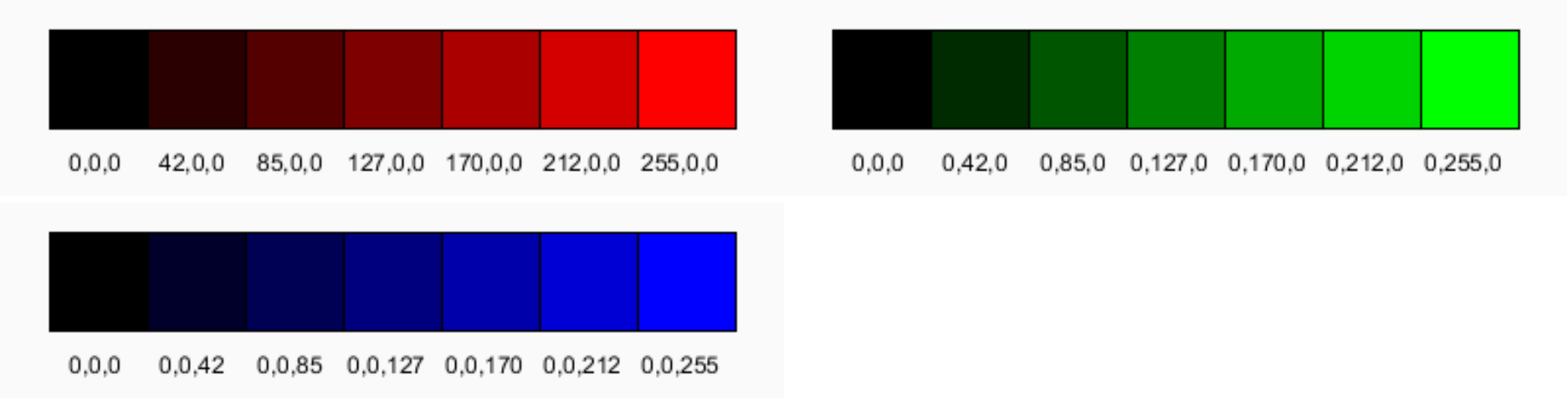

```
// background color to gray
background(127);

// setting the size of our canvas
size(10, 10);

// drawing a black rectangle with light gray border
stroke(250);
fill(0);
rect(3,3,5,4);

// drawing a light gray rectangle with black border
stroke(0);
fill(250);
rect(6,6,5,3);

```

`stroke(val)` sets the color of lines to `val` in grayscale

`fill(val)` sets the color inside a shape to `val` in grayscale

`background(val)` sets the color of the background to `val` in grayscale

Color



`stroke(r,g,b)` sets the color of lines to `r,g,b` in the red-green-blue scale

`fill(r,g,b)` sets the color inside a shape to `val` in the red-green-blue scale

`background(r,g,b)` sets the color of the background to `val` in red-green-blue scale

Interaction

Making things animated and interactive

In Processing this is managed by two methods

- The `setup` method specifies what will happen *at the beginning of the program, and will be executed only once*
- The `draw` method specifies what will happen *continuously, during your program's execution*. This is known as the *drawing loop*.

```
    // here we define global variables
    int wid = 600;
    int hei = 600;

    int x = wid;
    int y = hei;

    void setup(){
        // this only gets executed once, at the beginning of your program
        size(wid,hei);
    }

    void draw(){
        // this sets the background color
        background(100);

        // this gets executed continuously
        ellipse(x,y,100,100);
    }

```

Interaction

Making things animated and interactive

In Processing this is managed by two methods

- The `setup` method specifies what will happen *at the beginning of the program, and will be executed only once*
- The `draw` method specifies what will happen *continuously, during your program's execution*. This is known as the *drawing loop*.

```
    // here we define global variables
    int wid = 600;
    int hei = 600;

    int x = wid;
    int y = hei;

    void setup(){
        // this only gets executed once, at the beginning of your program
        size(wid,hei);
    }

    void draw(){
        // this sets the background color
        background(100);

        // this updates the position of the ellipse for the next frame
        x+=10;
        y=x*x/600;
        if (x >= 650){
            x = 0;
        }

        // this gets executed continuously
        ellipse(x,y,100,100);
    }

```

The speed of the drawing loop can be set with the `frameRate(fps)`, where `fps` is the desired number of *frames per second*

Interaction

Making things animated and interactive

The simplest kind of interaction that we get from Processing is keyboard and mouse input

- The `mousePressed` global variable is a boolean that is set to `true` when the user clicks a mouse button. Before every call to the `draw` method it is set to `false` by default.

- The global variables `mouseX` and `mouseY` give you the *pixel coordintates of the mouse pointer* in the canvas. These are of type `int`.

```
    // here we define global variables
    int wid = 600;
    int hei = 600;

    int x = wid/2;
    int y = hei/2;

    void setup(){
        // this only gets executed once, at the beginning of your program
        size(wid,hei);
    }

    void draw(){
        // this sets the background color
        background(100);

        // this gets executed continuously
        ellipse(mouseX,mouseY,100,100);
    }

```

Interaction

Making things animated and interactive

We can manage all the key presses outside the drawing loop

- The `keyPressed` method gets executed whenever the user presses a key
- You can get the ASCII value of the key that was pressed using the global variable `key`, of type `char`

```
// here we define global variables
int wid = 600;
int hei = 600;

int x = wid/2;
int y = hei/2;

int diameter = 100;

void setup(){
    // this only gets executed once, at the beginning of your program
    size(wid,hei);
}

void draw(){
    // this sets the background color
    background(100);

    // this gets executed continuously
    ellipse(mouseX,mouseY,diameter,diameter);
}

void keyPressed(){
    // if the key pressed is a, increase the diameter by 10
    // if the key pressed is b, decrease the diameter by 5
    if (key == 'a' || key == 'A' ){
        diameter += 10;
    } else if ( key == 'b' || key == 'B' ){
        diameter -= 5;
    }
}

```

Using Objects and Classes in Processing

Processing is based on Java, so we can use all of the things we learned during the COMP 202. Let's create a program that draws robots with the following rules:

- A `Robot` should be drawn as an ellipse. It has a position on the screen in pixel coordinates (`x`,`y`), and a `diameter` attribute. It will also have a `speed` attribute (in pixels per second).
- Every time the user clicks on the window, a new `Robot` will be drawn at the location of the click.
- Every time the user presses x, a `Robot` will be selected at random, and killed.
- `Robot` instances are scared of the last rule, so they should be *shaking* around their current location

Some additional rules

- All the `Robot` instances are `attracted` towards the location of the mouse pointer.
- `Robot` instances hate each other: add a `repelling` force between `Robot` instances that keeps them apart.

Next class

Now that we can draw stuff on a screen and animate it, we're going to:

- Create a couple more objects
- Load properties from a file
- Use exceptions to catch errors
- Learn a bit about physics based animation, probability, AI and robotics with our setup

Resources

- Processing:
  http://processing.org
- **Very** simple Processing Tutorial:
  http://hello.processing.org
- A book with **far more advanced** programming concepts on Processing:
  http://natureofcode.com/book/