Using Processing with Java

Grupo de Linux da Universidade de Aveiro

Eduardo Sousa
eduardosousa@ua.pt

University of Aveiro

2016

Code Structure

```
public class Teste extends PApplet {
    public static void main(String[] args) {
        PApplet.main(Teste.class.getName());
    }

    public void settings() {...}

    public void setup() {...}

    public void draw() {...}
}
```

What is the canvas?

The canvas is the window where all the objects will be drawn.

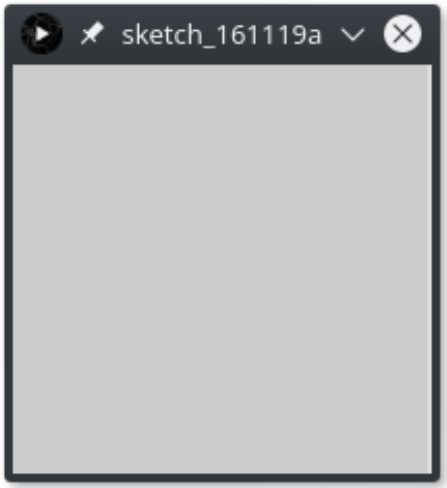


Figure: Canvas

Canvas coordinate system

This is the canvas coordinate system.
Objects will only be drawn if they are inside the canvas, e.g. *rect(1, 2, 2, 2)*.

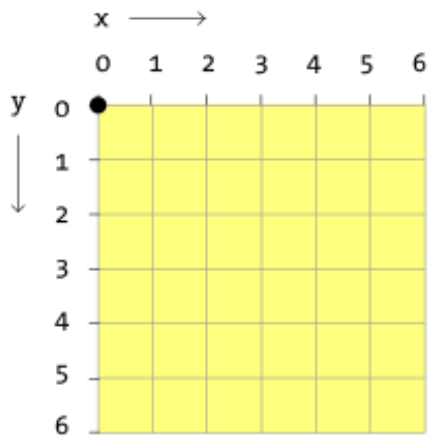


Figure: Canvas coordinate system

Canvas functions

Functions needed to alter the canvas:

```
// Can only be used in settings()
size(width, heigth);

// Cannot be used in settings()
// 0 <= grey, r, g, b <= 255
background(grey);
background(r, g, b);
```

Drawing Loop

In Processing, the drawing loop is based in frames. Each time a new frame is going to be presented in the display, Processing will call the function **draw()**.
The default frame rate is 60 frames per second, so in this case, every second the function **draw()** will be called 60 times.

Stoping drawing loop

To stop the the drawing loop, use **noLoop()**.
This will freeze the moving objects in the canvas.
Can be useful for pausing a game.

Drawing a new frame

If the drawing loop is stopped, you can use **redraw()** to render a new frame, without restarting the drawing loop.
Can be useful to draw a menu when the game is paused.

Resuming drawing loop

If the drawing loop is stopped, you can use **loop()** to restart the drawing loop.
Can be useful when you are resuming the game after a pause.

Drawing instruction order

For every object you are going to draw, there is an order to call the drawing functions.

1. Stroke/Fill
2. Mode
3. Object

Stroke

Stroke is the border line of the objects. It can be defined using the following functions.

```
// No border
noStroke();

// Size of line - default = 1
strokeWeight(weight);

// alpha == transparency
// 0 <= grey, r, g, b, alpha <= 255
stroke(grey);
stroke(r, g, b);
stroke(r, g, b, alpha);
```

Fill

Fill is the color of the object. It can be defined using the following functions.

```
// No fill
noFill();

// alpha == transparency
// 0 <= grey, r, g, b, alpha <= 255
fill(grey);
fill(r, g, b);
fill(r, g, b, alpha);
```

Point

To draw a point, use the function **point(x, y)**.

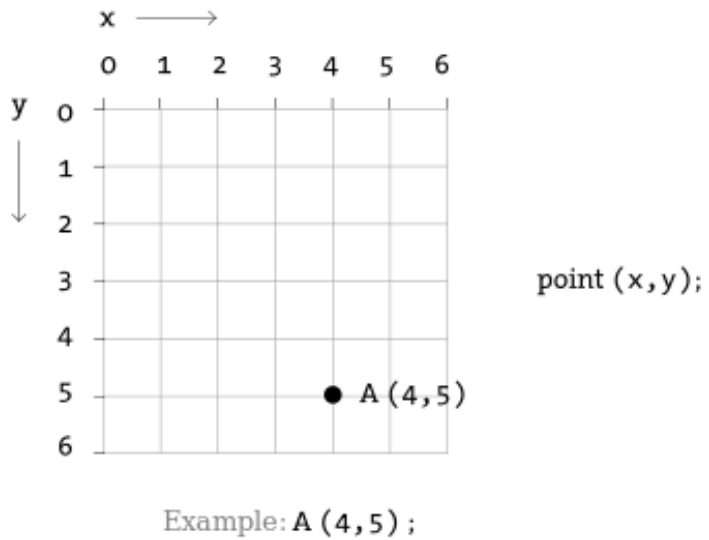


Figure: Point

Line

To draw a line, use the function **line(x1, y1, x2, y2)**.

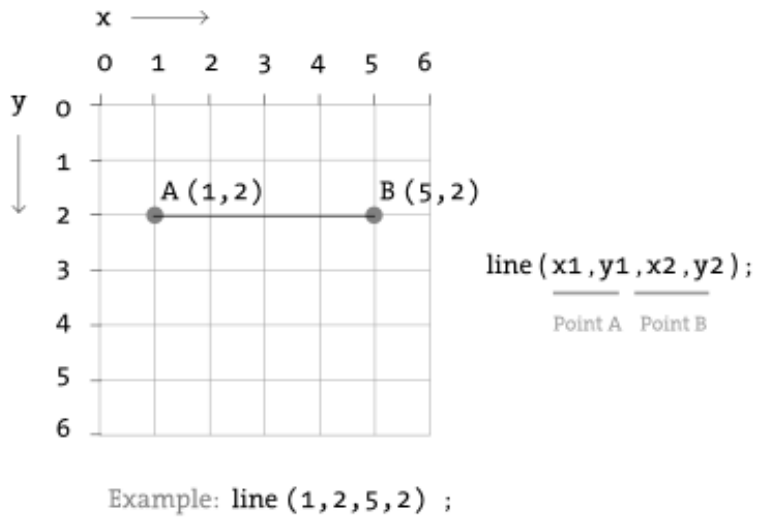


Figure: Line

Rectangle - RectMode = CORNER

To draw a rectangle, use the function **rect(x, y, width, heigth)**. The default mode for drawing a rectangle is the CORNER, where the upper left corner will be used as the reference point.

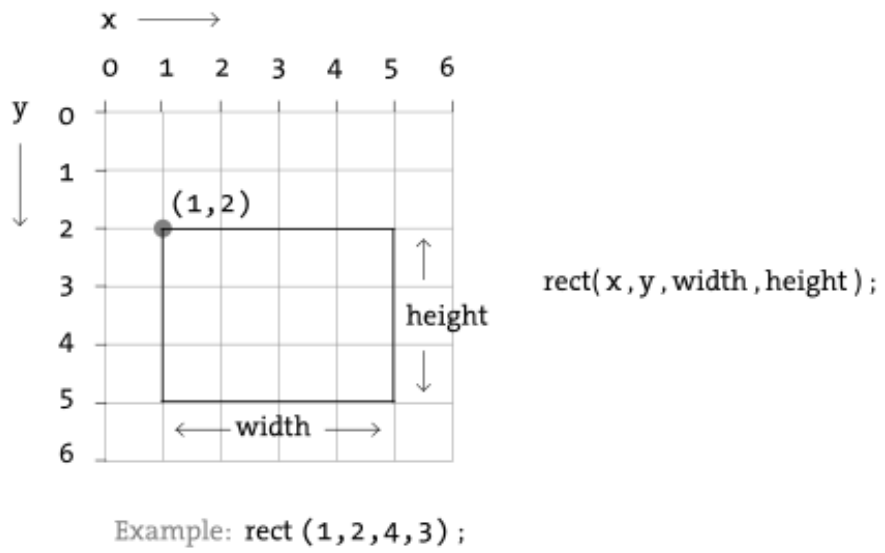


Figure: Rectangle - CORNER mode

Rectangle - RectMode = CENTER

To draw a rectangle, where the reference point is the center, use the functions as defined below:

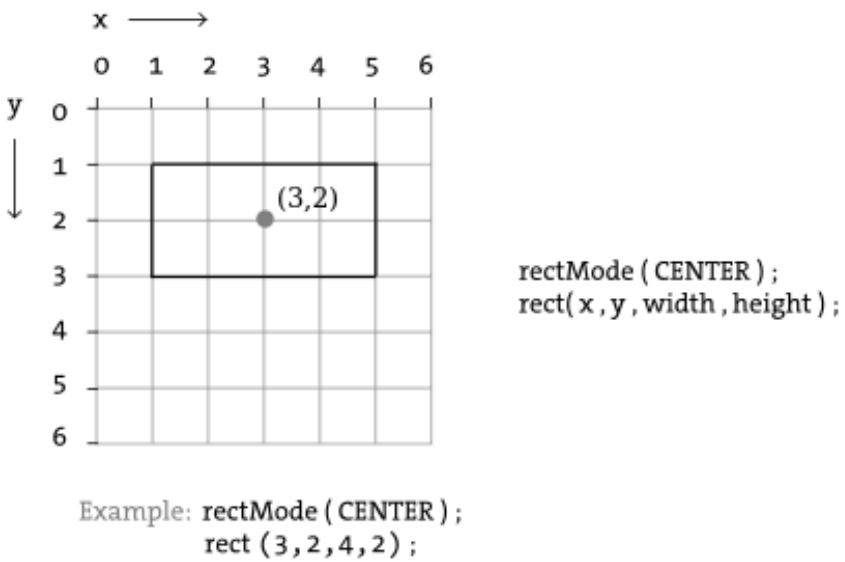


Figure: Rectangle - CENTER mode

Rectangle - RectMode = CORNERS

To draw a rectangle, where the reference points are upper left corner and lower right corner, use the function as defined below:

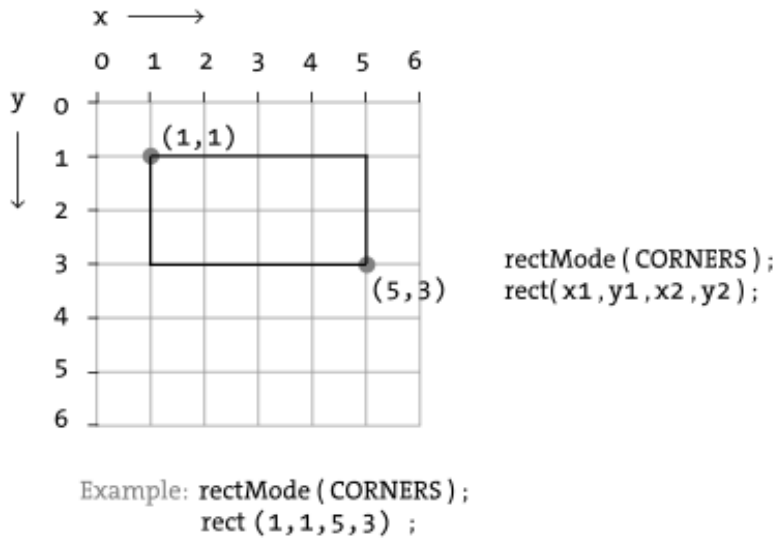


Figure: Rectangle - CORNERS mode

Ellipse - EllipseMode = CENTER

To draw an ellipse, use the function **ellipse(x, y, width, heigth)**. The default mode for drawing an ellipse is the CENTER, where the center will be used as the reference point.

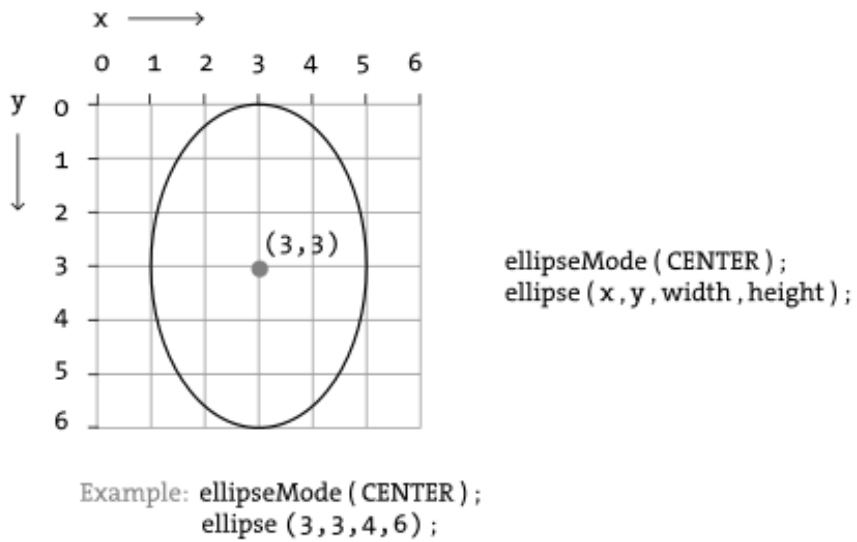
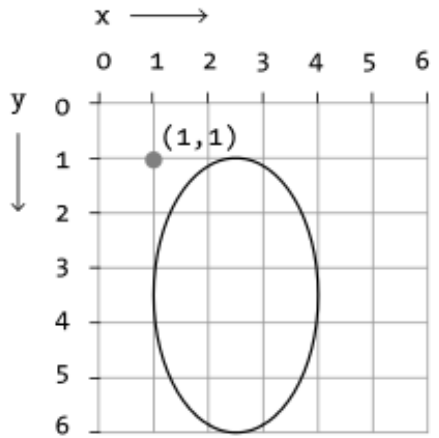


Figure: Ellipse - CENTER mode

Ellipse - EllipseMode = CORNER

To draw an ellipse, where the reference point is the upper left corner, use the functions as defined below:




```
ellipseMode ( CORNER ) ;
ellipse ( x , y , width , height ) ;
```

Figure: Ellipse - CORNER mode

Ellipse - EllipseMode = CORNERS

To draw an ellipse, where the reference points are the upper left coner and the lower right corner, use the functions as defined below:

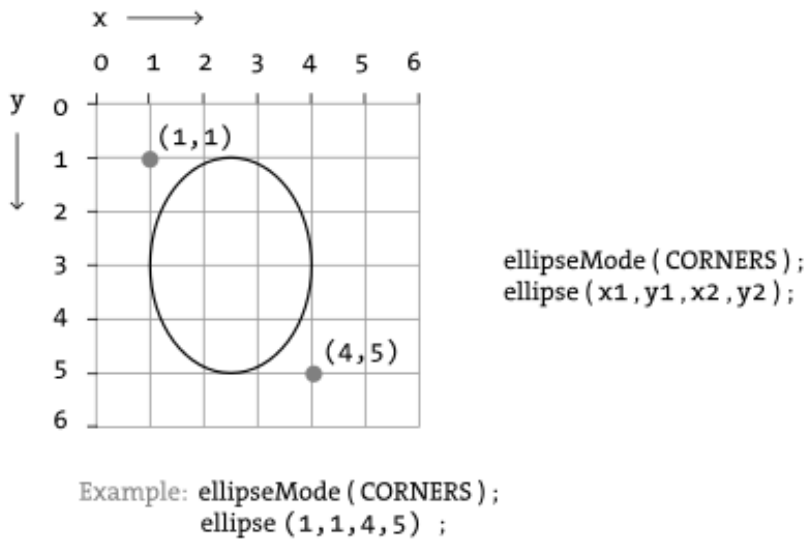


Figure: Ellipse - CORNERS mode

Using text

```
PFont f;

// Font, Size, Anti-aliasing
f = createFont("Arial", 16, true);
textFont(f);

// Specifying the color
fill(gray);

// Writing the text
text("Hello world!", x, y);
text("Hello world!", x1, y1, x2, y2);
```

Using images

To use images, use the following code:

```
// Declaring the image object
PImage img;

// Loading the image
img = loadImage("img.jpg");

// Showing the image
image(img, x, y);

// Resizing and showing the image
image(img, x, y, width, heigth);
```

It supports the following formats: GIF, JPG, TGA, PNG.

Using sound

To use sounds, use the following code:

```
import ddf.minim.*;

Minim minim = new Minim(this);

// Loading the sound file
AudioPlayer sound = minim.loadFile("song.mp3"

// Playing the sound
sound.play();

// Playing the sound file in a loop
sound.loop();
```

Using the mouse variables

In Processing, the mouse can only be used when the pointer is over the canvas.
There are some variables that are useful when using the mouse:

- **mouseX** - current X coordinate of the mouse pointer
- **mouseY** - current Y coordinate of the mouse pointer
- **pmouseX** - previous X coordinate of the mouse pointer
- **pmouseY** - previous Y coordinate of the mouse pointer
- **mousePressed** - true if the mouse is currently being pressed
- **mouseButton** - indicates which button of the mouse is being pressed (LEFT or RIGHT)

Using mouse functions

If one of the mouse actions happens, one or more of the following functions will be called:

```
public void mousePressed() {...}
public void mouseReleased() {...}
public void mouseClicked() {...}
public void mouseDragged() {...}
public void mouseMoved() {...}
public void mouseWheel() {...}
```

Note: you don't need to implement all these functions. Implement the ones that are needed.

Using keyboard variables

There are few variables that hold values when a key is pressed. These are the variables:

- **keyPressed** - true if there is a key being pressed
- **key** - key value if keyPressed == true
- **keyCode** - if key == CODED, keyCode will hold the code

Coded Keys

If the key is not specified in the ASCII code, you will have to use the **keyCode** variable to check what is the value.
Here you the most used keys that are coded:

- UP
- DOWN
- LEFT
- RIGHT
- ALT
- SHIFT
- CONTROL

Using keyboard functions

If one of the keyboard actions happens, one or more of the following functions will be called:

```
public void keyPressed() {...}

public void keyReleased() {...}
```

Note: you don't need to implement all these functions. Implement the ones that are needed.

Coded keys - Example

Here is an example of the use of coded keys:

```
public void keyPressed() {
    if(key == 'p' || key == 'P') {
        pauseTheGame();
    } else if(key == CODED) {
        if(keyCode == UP) {
            goUp();
        } else if(keyCode == DOWN) {
            goDown();
        }
    }
}
```

References

If you need more specific information, check these links:

- `https://processing.org/reference/`
- `https://processing.org/reference/libraries/`
- `https://processing.org/tutorials/`
- `https://processing.org/examples/`