

# Graphics Output on the hsafx Console

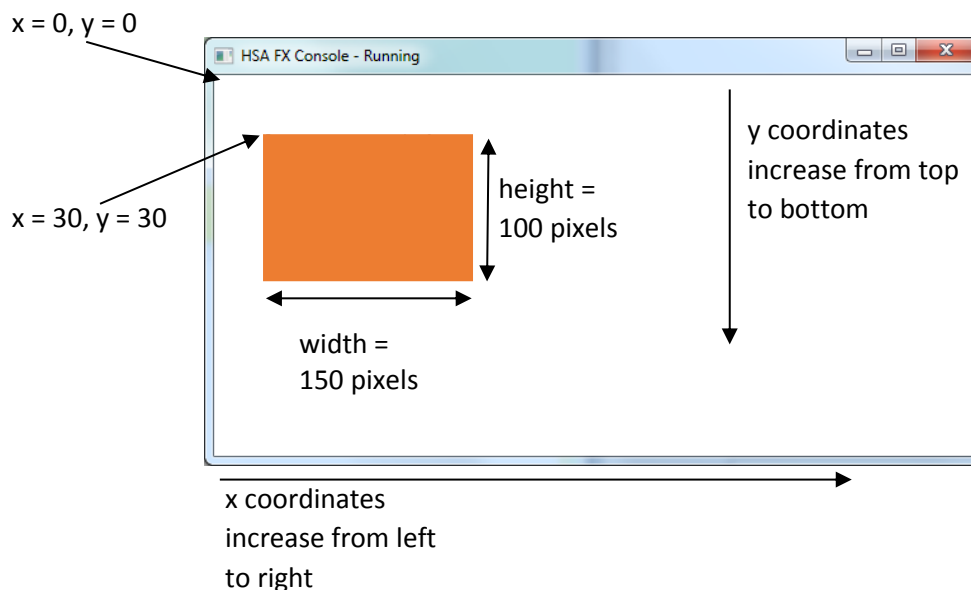
## Introduction

You already know that the Console window can output text, laid out in rows and columns of **characters** (any letter, number, punctuation mark, space, etc. is a character).

But the Console can also output graphics, and for that you have to think of it as laid out in rows and columns of **pixels** (a pixel is the smallest dot of color that can appear on a computer screen).

## Graphics Output

For graphics, the console is divided into **x and y coordinates**. Each x and y coordinate identifies a single pixel. The top left corner is x=0 and y=0. Whenever you draw a shape, you usually have to specify an x and y starting position, and at least a width and height for the shape, in pixels.



## Basic Graphics Methods

There are two kinds of drawing commands. Commands that start with "stroke" draw outlines, and commands that start with "fill" draw filled-in objects.

### **c.setFill("color")**

Sets the color for all "fill" drawing commands.

### **c.setStroke("color")**

Sets the color for all "stroke" drawing commands.

### **c.setLineWidth (width)**

Sets the width of the line for all "stroke" drawing commands.

**c.fillRect** (x, y, width, height)

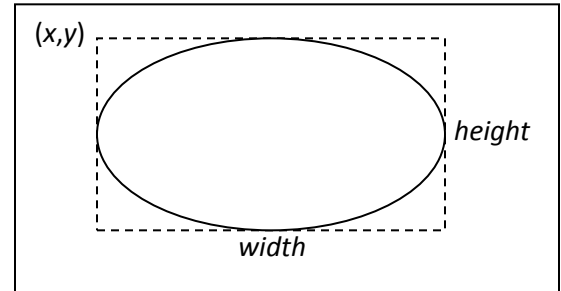
Draws a filled rectangle with upper-left corner at (x, y) with width of *width* and height of *height*.

**c.strokeRect** (x, y, width, height)

Draws a rectangle outline with upper-left corner at (x, y) with width of *width* and height of *height*.

**c.fillOval** (x, y, width, height)

Draws a filled oval. The oval is inscribed in the rectangle defined by the upper-left corner (x, y) with width of *width* and height of *height*.

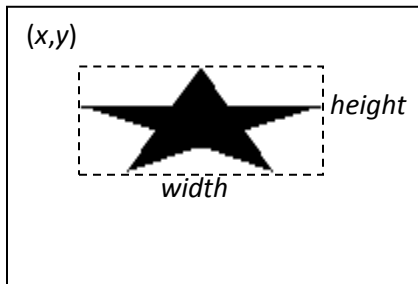


**c.strokeOval** (x, y, width, height)

Draws an oval outline. The oval is inscribed in the rectangle defined by the upper-left corner (x, y) with width of *width* and height of *height*.

**c.strokeLine** (x1, y1, x2, y2)

Draws a line from (x1, y1) to (x2, y2).



**c.fillStar** (x, y, width, height)

Draws a filled star. The star is inscribed in the rectangle defined by the upper-left corner (x, y) and the given *width* and *height*.

**c.strokeStar** (x, y, width, height)

Draws a star outline. The star is inscribed in the rectangle defined by the upper-left corner (x, y) and the given *width* and *height*.

**c.fillMapleLeaf** (x, y, width, height)

Draws a filled maple leaf. The leaf is inscribed in the rectangle defined by the upper-left corner (x, y) and the given *width* and *height*.

**c.strokeMapleLeaf** (x, y, width, height)

Draws a maple leaf outline. The maple leaf is inscribed in the rectangle defined by the upper-left corner (x, y) with width of *width* and height of *height*.

## Better Text Output

**c.fillText** (str, x, y)

Puts the string *str* at the starting point (x, y). The y coordinate is the bottom of the text.

**c.strokeText** (str, x, y)

Puts the string *str* at the starting point (x, y) but using outlines of the letters. The y coordinate is the bottom of the text.

### **c.setFont** ("font name", size)

Sets the font and font *size* for fillText and strokeText (but not for the print or println methods). The *font name* can be any font that's installed on your machine. Here are some fonts that are installed on most machines:

- Arial
- Arial Black
- Comic Sans MS
- Courier
- Georgia
- Impact
- Lucida Console
- Tahoma
- Times New Roman
- Trebuchet MS
- Verdana

### Advanced Graphics Methods (optional)

#### **c.fillRoundRect** (x, y, width, height, arcWidth, arcHeight)

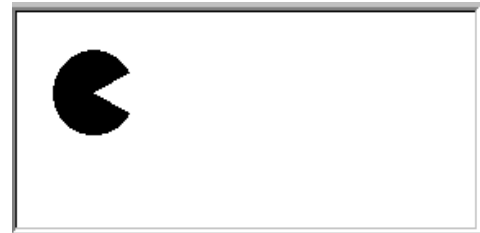
Draws a filled rectangle with rounded corners with upper-left corner at (x, y) with width of *width* and height of *height*. *arcWidth* and *arcHeight* are the width and height of the ellipse used to draw the rounded corners.

#### **c.strokeRoundRect** (x, y, width, height, arcWidth, arcHeight)

Draws a rectangle outline with rounded corners with upper-left corner at (x, y) with width of *width* and height of *height*. *arcWidth* and *arcHeight* are the width and height of the ellipse used to draw the rounded corners.

#### **c.fillArc** (x, y, width, height, startAngle, arcAngle)

Draws an arc. The arc is inscribed in the rectangle defined by the upper-left corner (x, y) with width of *width* and height of *height*. It starts at *startAngle* degrees and goes counterclockwise for *arcAngle* degrees.



```
c.fillArc (20, 20, 50, 50, 30, 300);
```

#### **c.strokeArc** (x, y, width, height, startAngle, arcAngle)

Draws an arc outline. The arc is inscribed in the rectangle defined by the upper-left corner (x, y) with width of *width* and height of *height*. It starts at *startAngle* degrees and goes counterclockwise for *arcAngle* degrees.

#### **c.setFill**("rgba(red, green, blue, alpha)")

Sets a color using *red*, *green*, and *blue* values with an alpha transparency. When alpha is 1, the color is solid. When the alpha is 0, the color is totally transparent. 0.5 is half transparent, etc. Also works with c.setStroke() and c.setBackground().

### Advice: Work Small

**Keep the console window small if you can.** The console does a lot of work in the background to make things easier for you. But if you use a very large console (e.g. 1200 x 900 pixels) and do a lot of drawing, this background work can cause the app to bog down and produce memory ("Java Heap Space") errors.

**See the handout in the "Extra" section** if you really need a large window, or if you are having memory error issues even on a small screen.