

# Programming 01

Environment

Drawing

Syntax

File handling

Variables

Loops

Conditionals

UIC/F08/AD508/Sauter

based on [processing.org](http://processing.org) and Casey Reas, Ben Fry: Processing: a programming handbook for visual designers and artists. MIT Press. Forthcoming in Fall 07

Environment

# Environment + Toolbar

The GUI consists of a text editor for writing code, message area, console, menu, toolbar, tabs for managing files

**Run:** Compiles the code, opens a display window, and runs the program inside.

**Stop:** Terminates a running program, but does not close the display window.

**New:** Creates a new sketch.

**Open:** Provides a menu with options to open files from the sketchbook, open an example, or open a sketch from anywhere on your computer or network.

**Save:** Saves the current sketch to its current location. If you want to give the sketch a different name, select “Save As” from the File menu.

**Export:** Exports the current sketch as a Java applet embedded in an HTML file. The folder containing the files is opened. Click ocomputer's default web browser.

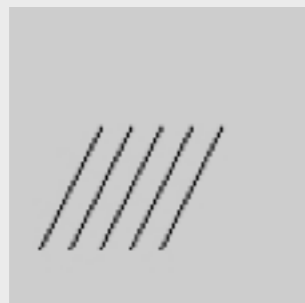
# Sketches

All Processing projects are called sketches

Each sketch has its own folder (you can use the option in the menu Sketch/Show Sketch Folder to show the content of this folder)

Example sketch:

```
line(10, 80, 30, 40); // Left line  
line(20, 80, 40, 40);  
line(30, 80, 50, 40); // Middle line  
line(40, 80, 60, 40);  
line(50, 80, 70, 40); // Right line
```



# Coordinates

The size of the display window is controlled with the `size()` function:

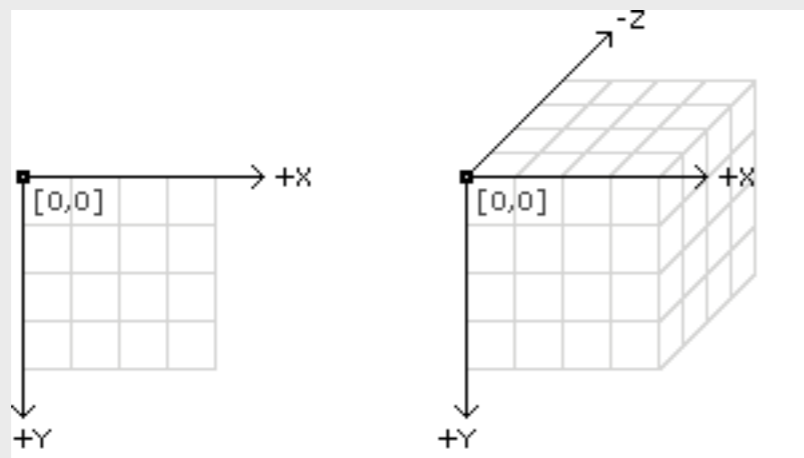
```
size(width, height)
```

The size function has two parameters: the first sets the width of the window and the second sets its height.

A position on the screen is comprised of an x-coordinate and a y-coordinate.

The x-coordinate is the horizontal distance from the origin and the y-coordinate is the vertical distance.

In a 400 pixel wide by 400 high window, `[0, 0]` is the upper-left pixel, `[320, 240]` is in the lower-right. The last visible pixel in the lower-right corner of the screen is at `[319, 239]`



# Syntax

Elements of the language and how they are used:

- Statement
- Statement Terminator
- Function
- Parameter
- Comment

```
//example statements
```

```
line(10, 80, 30, 40);
```

```
i = i + 1;
```

```
String txt = "Daniel";
```

```
print(txt);
```

note: Processing is case-sensitive: String, not string!

# Drawing

Defining the window size:

```
size(300, 300);
```

Setting the background color:

```
background(0); //black
```

```
background(255); //white
```

```
background(255, 0, 0); //red in rgb syntax
```

# Drawing Graphic Primitives

```
size(200, 200);  
noFill();  
  
//draws a point at x=10 and y=20  
point(10, 10);  
  
// draws a line from x1, y1 to x2, y2  
line(10, 10, 100, 100);  
  
// draws a rectangle over x1, y1, width, height  
rect(10, 10, 180, 140);  
  
// draws an ellipse x, y, width, height  
ellipse(120, 50, 40, 40);  
  
// draws a four sided polygon x1, y1, x2, y2, x3, y3, x4, y4  
quad(38, 31, 86, 20, 69, 63, 30, 76);  
  
// draws a triangle x1, y1, x2, y2, x3, y3  
triangle(120, 120, 80, 160, 160, 180);
```

# Defining Stroke and Fill Color

```
stroke(120); // gray stroke
stroke(0, 0, 255); // blue stroke
stroke(200, 80); // gray fill, transparent
noStroke(); // no Stroke
fill(100); // gray fill
fill(255, 0, 0); // red fill
fill(0, 255, 0, 127); green fill semi transparent
noFill();
```

# strokeWeight() and smooth()

```
background(0); // Sets the black background
stroke(255); // Sets line value to white
strokeWeight(5); // Sets line width to 5 pixels
smooth(); // Makes lines with smooth edges
line(10, 80, 30, 40); // Left line
line(20, 80, 40, 40);
line(30, 80, 50, 40); // Middle line
line(40, 80, 60, 40);
line(50, 80, 70, 40); // Right line
```

example:

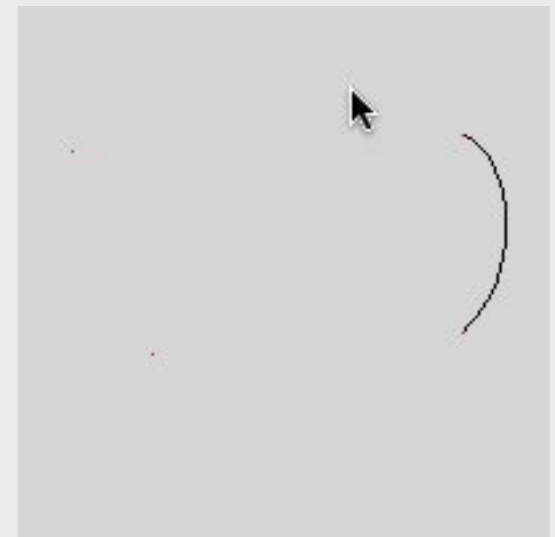
```
rect(10, 10, 50, 50);
fill(204); // Light gray
rect(20, 20, 50, 50);
fill(153); // Middle gray
rect(30, 30, 50, 50);
fill(102); // Dark gray
rect(40, 40, 50, 50);
```

# Curve Primitives

```
curve(10, 30, 90, 30, 60, 70, 20, 50);
```

curve draws a curve through 4 points, defined by 8 parameters, the curve is only visible between the second and third point (parameters 3-6) along the curve.

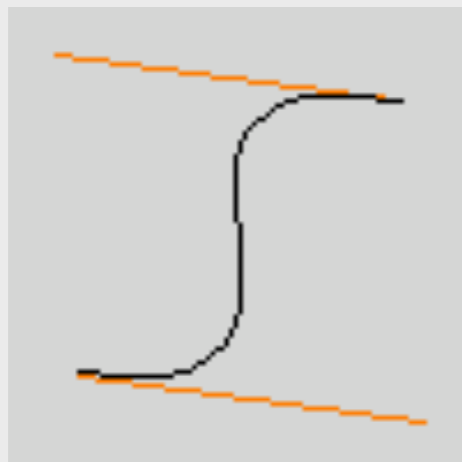
```
size(200, 200);  
curve(20, 54, 166, 48, 166, 122, 50, 130);  
stroke(255, 0, 0);  
point(20, 54);  
point(166, 48);  
point(166, 122);  
point(50, 130);
```



# Bezier Curves

Curve defined by a series of anchor and control points.  
The curve is drawn between the first and the last point and uses the two middle points to define the shape of the curve.  
Bezier curves were developed by French engineer Pierre Bezier.

```
stroke(255, 102, 0);  
line(85, 20, 10, 10);  
line(90, 90, 15, 80);  
stroke(0, 0, 0);  
bezier(85, 20, 10, 10, 90, 90, 15, 80);
```



# Variables

- Used to store values
- Have a **name** and a **value** (give names that are descriptive)
- Case-sensitive
- Name must not begin with special characters

```
String name = "Daniel"; // Declare and assign
int number = 32; // Declare and assign
int counter = 12; // Declare and assign
print(number);
print(name);
print(counter);
```

# Data Types

`int` //Integer: e.g. 1, 2, 3, ...

`float` //Floating point number: e.g. 0.1, 2.747, ...

`char` //Character: "\$", "A", stores one character.

`String` //String: e.g. "Daniel", series of characters.

`boolean` //Boolean: true or false.

```
int x;           // Declare the variable x of type int
float y;         // Declare the variable f of type float
boolean b;      // Declare the variable b of type boolean
x = 50;         // Assign the value 50 to x
y = 12.6;       // Assign the value 12.6 to f
b = true;       // Assign the value true to b
```

# Programming Modes

Basic Mode: This mode is used drawing static images and learning fundamentals of programming. Simple lines of code have a direct representation on the screen. The following example draws a yellow rectangle on the screen:

```
size(200, 200);  
background(255);  
noStroke();  
fill(255, 204, 0);  
rect(30, 20, 50, 50);
```

Continuous Mode: Adding more structure to a program opens further possibilities. The `setup()` and `draw()` functions make it possible for the program to run continuously – this is required to create animation and interactive programs.

```
void setup()  
{  
  size(200, 200);  
  noStroke();  
  background(255);  
  fill(0, 102, 153, 20);  
  smooth();  
}  
  
void draw()  
{  
  ellipse(mouseX, mouseY, 50, 50);  
}
```

# Mouse position

`mouseX`: X-Position the mouse within the applet

`mouseY`: Y-Position

```
line(mouseX, 20, mouseX, 80);
```

`mousePressed` returns true while mouse is pressed, false if not.

```
void draw() {  
    if (mousePressed == true) {  
        fill(0);  
    } else {  
        fill(255);  
    }  
    rect(25, 25, 50, 50);  
}
```

# Conditionals

Statements within the `if` section are only executed in case the condition (`i < 35`) is `true`; statements within the `else` section are executed only in case the condition is `false`.

```
void setup () {  
    size(300, 300);  
}  
void draw() {  
    if(mouseX < 150) {  
        line( 0, mouseY, 150, mouseY );  
    }  
    else {  
        line( 150, mouseY, 300, mouseY );  
    }  
}
```

# Relational Operators

Used to compare values (conditionals):

> (greater than)  
< (less than)  
>= (greater than or equal to)  
<= (less than or equal to)  
!= (inequality)  
== (equality)

```
5 > 4      // True
5 < 3      // False
5 > 5      // False
5 >= 5     // True
5 >= 6     // False
5 != 5     // False (not equal)
5 == 5     // True
5 == 4     // False
```

# Loops

The `for()` loop uses defined conditions

```
for (int i=40; i<80; i=i+5) {  
    line(30, i, 80, i);  
}
```

The `while()` loop repeats as long as the condition is true

```
int i=0;  
while (i<80) {  
    line(30, i, 80, i);  
    i = i+5;  
}
```

note: if the test condition in the while loop cannot be false, the program freezes

Form 00: Draw three lines and tree points

Form 01: Draw tree rectangles and one ellipse.

Form 02: Draw four quads

Form 03:

Control the position of two lines with one variable.

Form 04: Control the position and size of two lines with two variables.

Form 05: Control the properties of two shapes of your choice with two variables.

Form 06: Redo Exercise Form 04 using mouseX and mouseY as the variables.

Form 07: Draw three visual elements that each move in relation to the mouse in a different way.

Form 08: Create a simple, regular pattern with eight curves.

Form 09: Create a Bezier curve with its curvature being controlled by the vertical position of the mouse.

Form 10: Draw a layered form with two loops and at more than 20 elements.

Form 11: Move a visual element across the screen. When it disappears off the edge, move it back into the frame.

Form 12: Draw a visual element that moves in relation to the mouse, but with a different relation when the mouse is pressed.

Form 13: Using if() and else(), make the mouse perform different actions when in different parts of the window.

# File handling and exporting

Remember: Each sketch has its own directory/folder where the main program file is located with the ending .pde

You can browse to this folder by choosing Sketch/Show Sketch folder from the Processing menu.

Example:

Sketch name: "Sketch\_01",

The directory for the sketch will be called "Sketch\_01"

The main file will be called "Sketch\_01.pde".

If you click on 'export' in the Processing menu bar, a subfolder called 'applet' will be created containing all files necessary to publish your work.

the .jar file is an archive containing the Java applet which you need to upload your work on the class website.

a index.html file is included which will show your applet in the Web browser.

You can see the content of your actual sketch folder by choosing 'Sketch/Show sketch folder' in the Processing menu bar.

# Language Reference

<http://processing.org/reference>

There are many examples in the learning section of the processing website:

<http://processing.org/learning>

If you click 'open' the Processing menu bar you will find a folder called 'examples' with a variety example programs.