LEARNING processing

#processing

Table of Contents

About

Chapter 1: Getting started with processing

Remarks

Processing is an open source programming language and environment for people who want to create images, animations, and interactions.

Processing refers to the language built on top of Java and the minimal IDE it ships with. It is free and open-source, runs on Linux, Mac OS X, and Windows, and can output for screens, print, 3D packages and CNC printing.

The language simplifies a lot of complex concepts and eases the entry of designers, artists and non-programmers to the world of programming.

Over the years it was used to produce a number of projects ranging from data visualization, to physical computing , games, 3D, sound, live perfomance, and more.

Due to its vibrant community, Processing not only enjoys a contribution of over 100 libraries, but is also present on major mobile platforms such as Android and iOS.

There are online communities for sharing Processing content, like OpenProcessing.

Some websites even allow users to learn and use Processing directly in the browser, like the Flash-driven SketchPatch and the JavaScript-driven HasCanvas,Sketchpad and p5.js(pure JS).

There are also Processing ports to the following languages:

- JavaScript using ProcessingJS or p5js
- ActionScript
- Python (see NodeBox, Field, pyProcessing or the new official Python Mode)
- Scala
- Clojure
- Ruby

The Android mode allows to run Processing sketches as Android applications with little or no changes in the code by automating tasks from project setup to *.apk* file export. Android Processing sketches also have access to the underlying Android sensors and devices.

Advanced users are not constrained to the Processing IDE; they can set up Processing projects in Eclipse; use proclipsing or alternatively use Sublime Text to build and run sketch via the processing-sublime package.

Versions

Version	Release Date
1.5.1	2011-05-15

Version	Release Date
2.2.1	2014-05-19
3.1.2	2016-07-29
3.2.1	2016-08-19

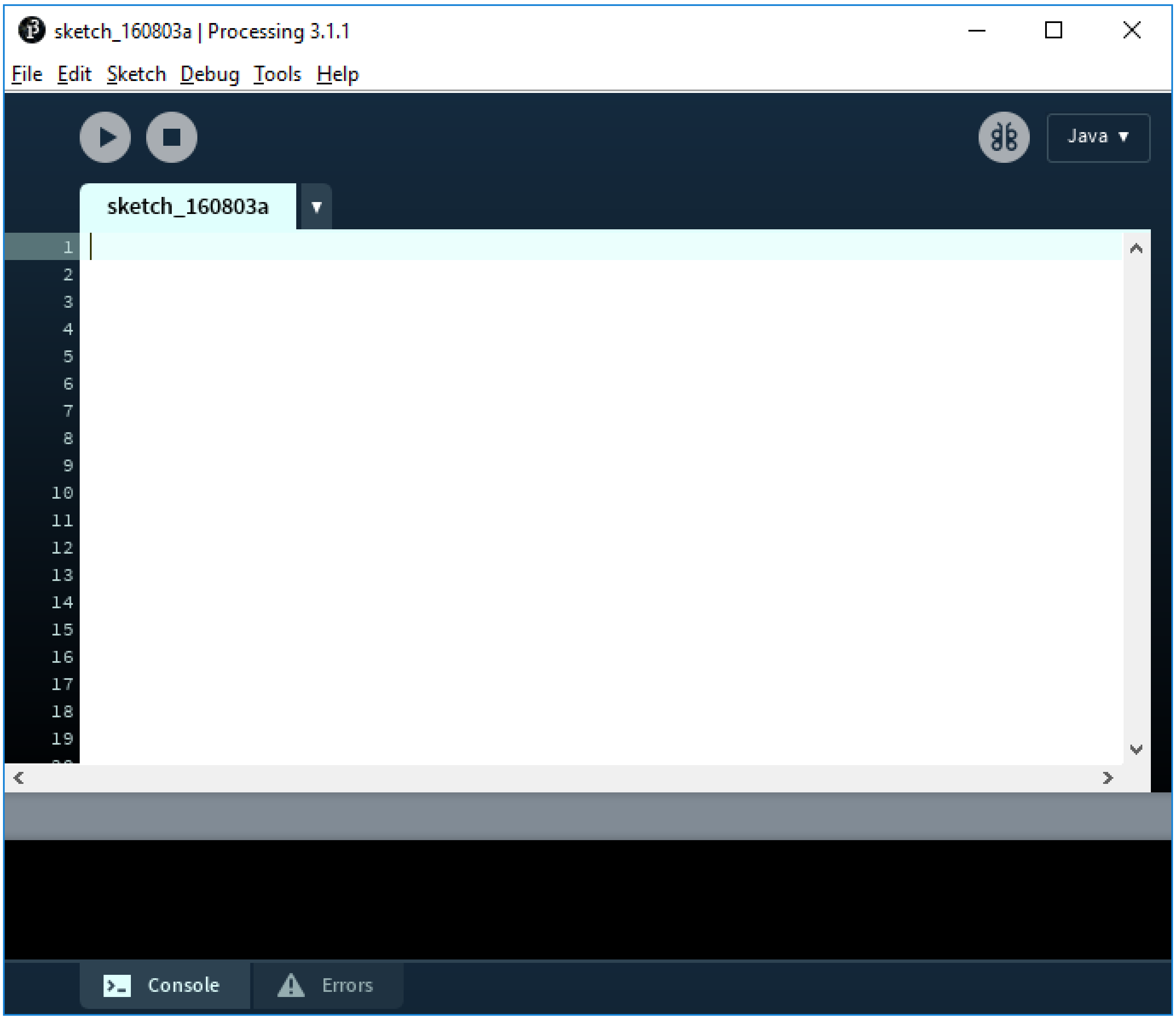
Examples

Installation and Setup

The easiest way to use Processing is by downloading the Processing editor from the Processing download page.

That comes as a zip file. Unzip that file anywhere, and you'll have a directory that contains an executable (on Windows, that's `processing.exe`).

Running that executable opens up the Processing editor:


sketch_160803a | Processing 3.1.1
File Edit Sketch Debug Tools Help
Java
sketch_160803a
Console
Errors

The Processing editor (also called the Processing Development Environment, or PDE) contains many tools that do a lot of work for you. It allows you to write Processing code, which it automatically converts to Java and then compiles and runs for you.

The PDE contains many features, but for now just write your Processing code inside the white section of the editor, and then press the play button to run your code. See the Hello World section below for some example code.

You can also write Processing code using other basic code editors like Atom or Sublime Text, or with a more advanced IDE like eclipse.

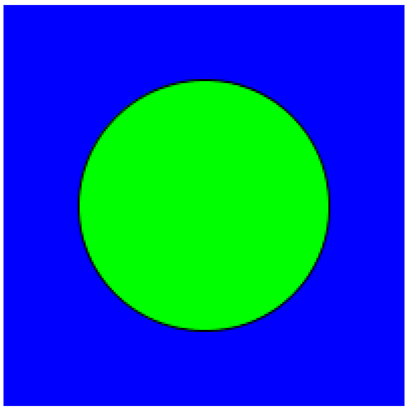
Hello World

The easiest way to write Processing code is to simply call a series of functions. Press the run button in the Processing editor, and Processing will run your code. Here's an example:

```
size(200, 200);
background(0, 0, 255);
fill(0, 255, 0);
```

```
ellipse(100, 100, 100, 100);
```

This code creates a `200x200` window, draws a blue background, changes the fill color to green, and then draws a circle in the middle of the screen.



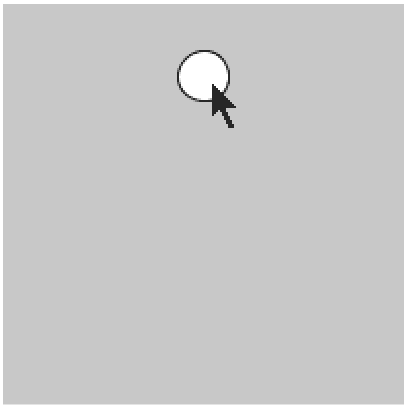
However, most Processing sketches will use the predefined `setup()` and `draw()` functions.

- The `setup()` function is called automatically by Processing, once at the very beginning of the sketch. This function is used for doing the initial setup, such as `size`, and loading of resources such as image and sound files.
- The `draw()` function is called automatically by Processing 60 times per second. This function is used for drawing and getting user input.

  ```
  void setup() {
    size(200, 200);
  }

  void draw(){
    background(0);
    ellipse(mouseX, mouseY, 25, 25);
  }
  ```

This code creates a `200x200` window and then draws a circle at the current mouse position.



Read Getting started with processing online: https://riptutorial.com/processing/topic/3538/getting-started-with-processing

Chapter 2: Basic shapes and functions using P3D

Syntax

- translate(float x, float y, float z)
- rotateX(float angle)
- rotateY(float angle)
- rotateZ(float angle)
- box(float size)
- box(float w, float h, float d)

Parameters

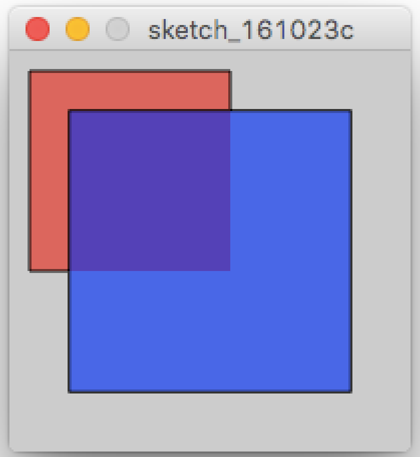
Parameters	Details
angle	the angle is in radians
size	the dimension of the box to be used for all its dimensions
w	the dimension of the box in the `x-axis`
h	the dimension of the box in the `y-axis`
d	the dimension of the box in the `z-axis`

Examples

3D Translation

Here's how to translate objects in P3D:

```
size(200, 200, P3D); //Starting P3D renderer
fill(255, 0, 0, 150); //transparent red
rect(10, 10, 100, 100); //first rectangle
fill(0, 0, 255, 150); //transparent blue
translate(50, 50, 50); //translate x, y and z by 50 pixels
rect(0, 0, 100, 100); //second rectangle (same dimensions as the first one)
```



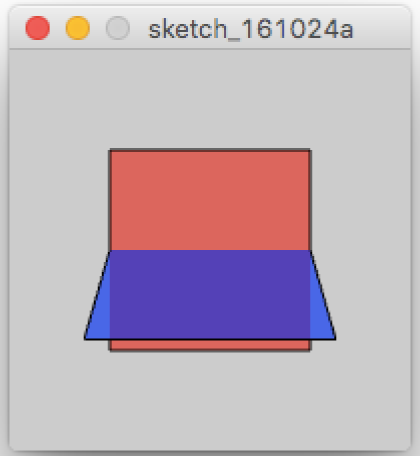
Red: first rectangle Blue: second rectangle

As can be seen from the above sketch, the second rectangle only appears to be larger than the first one, when in reality it is "closer" to the screen as a result of translating the rectangle `50` pixels along the `z-axis` (and of course, the rectangle has been translated along the `x` and `y` axes).

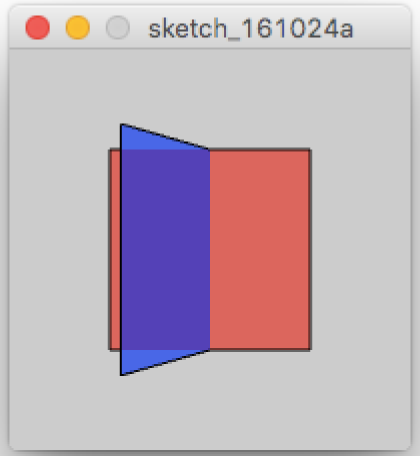
3D Rotation

There are three functions for 3D rotation: `rotateX(angle)`, `rotateY(angle)` and `rotateZ(angle)` for rotation in their respective axes where `angle` is in radians.

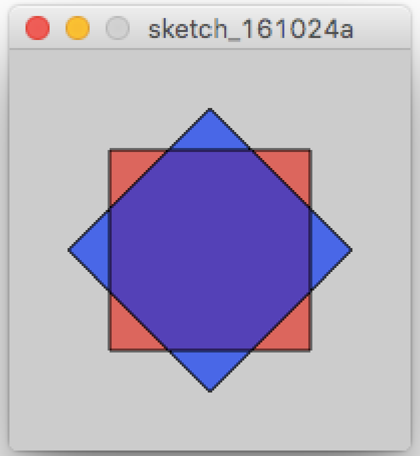
```
size(200, 200, P3D); //Starting P3D renderer
fill(255, 0, 0, 150); //transparent red
translate(width/2, height/2);//translate to centre, ie (100, 100)
rectMode(CENTER);//This makes the rectangle centre in (100, 100)
rect(0, 0, 100, 100); //first rectangle
fill(0, 0, 255, 150); //transparent blue
rotateX(PI/4); //rotate in the x-axis by PI/4 radians (45 degrees)
rect(0, 0, 100, 100); //second rectangle (same dimensions as the first one)
```



```
rotateY(radians(45)); //rotate in the y-axis by passing the radians conversion of 45 degrees
```



```
rotateZ(3*PI/4); //rotate in the z-axis by 3*PI/4 radians (270 degrees)
```

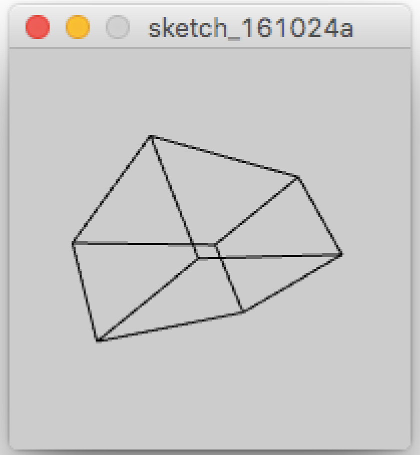


Note: transformations (such as translations and rotations) add on to the previous transformation.

Drawing a cuboid

To draw a cuboid, you have to use the `box()` function by giving its dimensions as its parameters.

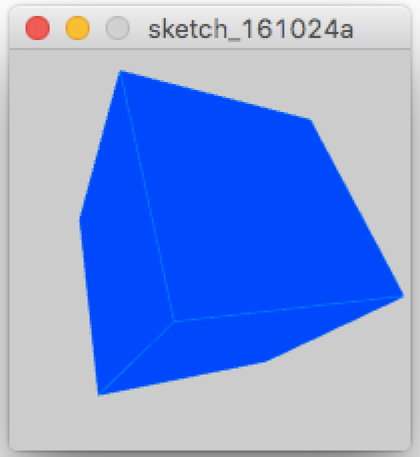
```
size(200, 200, P3D); //Starting the P3D renderer
translate(width/2, height/2); //Translating to the centre of the sketch
rotateY(PI/4); //rotate so that...
rotateX(PI/6); //... it will be easy to see the box
noFill(); //disabling the box's fill, so that we will be able to see its edges
box(100, 50, 75); //the box function requires its dimensions as its parameters
```



Note that the `box()` function does *not* accept its position as the parameters

There also is a way to call the `box()` function with only one parameter. In this case, it will be a cube.

```
stroke(0, 100, 255); //change the edges' colour
fill(0, 0, 255); //fill the `box` in a blue colour
box(100); //draw a cube
```



Read Basic shapes and functions using P3D online: https://riptutorial.com/processing/topic/7591/basic-shapes-and-functions-using-p3d

Chapter 3: Colours in Processing

Introduction

This article will show you the various colours formats in Processing and the ways in which they can be used.

Syntax

- color(r, g, b);
- color(r, g, b, alpha);
- color(gray);
- color(gray, alpha);
- color(h, s, l); //The mode must be HSB. You can change this using colorMode.

Parameters

Parameters	Details
r	Is the red of the color when the mode is `RGB`.
g	Is the green of the color when the mode is `RGB`.
b	Is the blue of the color when the mode is `RGB`.
alpha	Is the opacity of the color.
h	The hue of the color when the mode is `HSB`.
s	The saturation of the color when the mode is `HSB`.
l	The brightness/lightness of the color when the mode is `HSB`.
gray	The value between black (being 0) and white (being 255).

Remarks

Although it has not been mentioned in the official Processing documentation, there is a `CMYK` mode which you can use.

Examples

Colour Notation

There are various ways to use colours in Processing since Processing is very flexible with colour formats.

RGB and RGBA

This is the standard RGB(A) notation and the default color mode. The first three colour values (red, green, blue) range from `0` to `255`. For example, the below example is the colour red since red is maxed out at `255` while the others colours are at `0`. White is at `(255, 255, 255)` and black is `(0, 0, 0)`. The optional 4th parameter indicates the alpha value -- i.e. transparency. As in other the components, the range of values is again [0-255]; `0` being completely transparent and `255` being completely solid.

```
color(255, 0, 0) // This is red

color(0, 255, 0, 255) // This is opaque green, and is the same as color(0, 255, 0)

color(255, 255, 0, 10) // This is almost transparent yellow
```

HSB

HSB notation is similar to RGB notation, except for the fact that red, green, and blue are replaced with hue, saturation, and brightness, respectively. You can switch into HSB by using `colorMode(HSB)`.

```
color(0, 0, 255) //This is white
```

As with RGB, HSB also has the alpha value as the fourth parameter.

Gray Values

If one parameter is specified to a color function, it will be interpreted as the amount between black and white. White is represented as 255, and black as 0. It is the same as `color(param1, param1, param1)` in RGB mode. If two parameters are specified, then the first parameter will be interpreted as above and the second will be the alpha value.

Read Colours in Processing online: https://riptutorial.com/processing/topic/8283/colours-in-processing

Chapter 4: Drawing Basic Shapes

Introduction

In Processing, drawing shapes is key to the program. Otherwise, nothing would appear on the screen. This section will tell you how basic shapes are drawn.

Syntax

- line(float x1, float y1, float x2, float y2)
- line(float x1, float y1, float z1, float x2, float y2, float z2)
- ellipse(float x, float y, float w, float h)
- rect(float x, float y, float w, float h)
- triangle(float x1, float y1, float x2, float y2, float x3, float y3)

Parameters

Parameter	Details
x1	x-coordinate of the first point
y1	y-coordinate of the first point
z1	z-coordinate of the first point
x2	x-coordinate of the second point
y2	y-coordinate of the second point
z2	z-coordinate of the second point
x3	x-coordinate of the third point
y3	y-coordinate of the third point
x	x-coordinate
y	y-coordinate
w	width
h	height

Remarks

You can find a reference on Processing's foundation.

Processing homepage

Examples

Drawing a Line

Processing provides a method named `line()` to draw a line on the screen. This code draws a white 10 pixel line on black background.

```
void setup() {
    size(500, 500);
    background(0);
    stroke(255);
    strokeWeight(10);
}

void draw() {
    line(0, 0, 500, 500);
}
```

The signature of method `line()` is this.

```
line(x1, y1, x2, y2);
```

`x1` and `y1` is a coordinate of the starting point. `x2` and `y2` is a coordinate of the ending point.

Method `stroke()` is used to specify the color of the line you will draw.

Method `strokeWeight()` is used to specify the thickness of the line you will draw. (in pixels)

Drawing a Rectangle

Processing provides method `rect()` to draw a rectangle. This code draws a white 50 X 50 rectangle on black background.

```
void setup() {
    size(500, 500);
    background(0);
    fill(255);
    noStroke();
}

void draw() {
    rect(225, 225, 50, 50);
}
```

The signature of method `rect()` is this.

```
rect(x, y, w, h);
```

`x` and `y` is the coordinate of the rectangle. `w` and `h` is rectangle's width and height.

Method `fill()` is used to specify the filling color of the rectangle and other shapes such as ellipse, triangle, polygon.

Method `noStroke()` is used to specify that that there are no strokes around the rectangle. This method also affects other shapes such as ellipse, triangle, polygon.

Drawing an Ellipse

Processing provides method `ellipse` in order to draw ellipse. This code draws a white circle which has radius of 25 pixels.

```
void setup() {
    size(500, 500);
    background(0);
    fill(255);
    noStroke();
}

void draw() {
    ellipse(225, 225, 50, 50);
}
```

The signature of method `ellipse()` is this.

```
ellipse(x, y, w, h);
```

`x` and `y` is the coordinate of the ellipse. `w` and `h` is ellipse's width and height.

Drawing a Triangle

Processing provides the method `triangle` in order to draw a triangle. The code below draws a nearly equilateral triangle of 25 pixels between each defining point.

```
void setup() {
    size(500, 500);
    background(0);
}
void draw() {
    triangle(0, 0, 25, 0, 12, 12);
}
```

The signature of `triangle` is as so:

```
triangle(x1, y1, x2, y2, x3, y3);
```

Each `x` point corresponds to the point's x axis, and `y` to the y axis. The three points will be joined to form a triangle.

Drawing a Triangle

Processing provides method `triangle()` to draw a triangle. This code draws a white triangle on black background.

```
void setup() {
    size(500, 500);
    background(0);
    fill(255);
    noStroke();
}

void draw() {
    triangle(250, 225, 225, 275, 275, 275);
}
```

Read Drawing Basic Shapes online: https://riptutorial.com/processing/topic/7277/drawing-basic-shapes

Chapter 5: Getting started with Processing for Android

Remarks

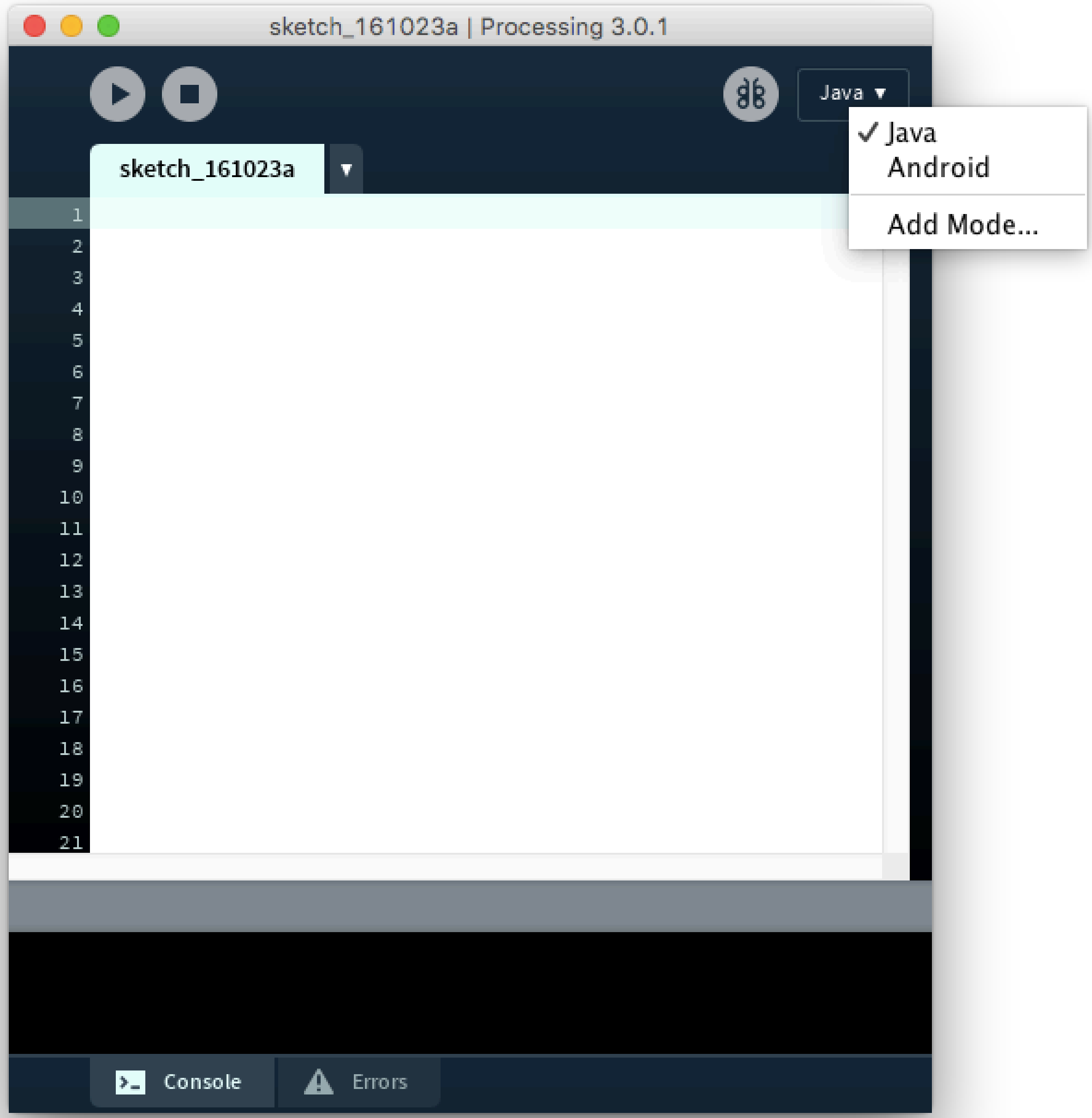
You can find more information on Processing for Android at http://android.processing.org/

Examples

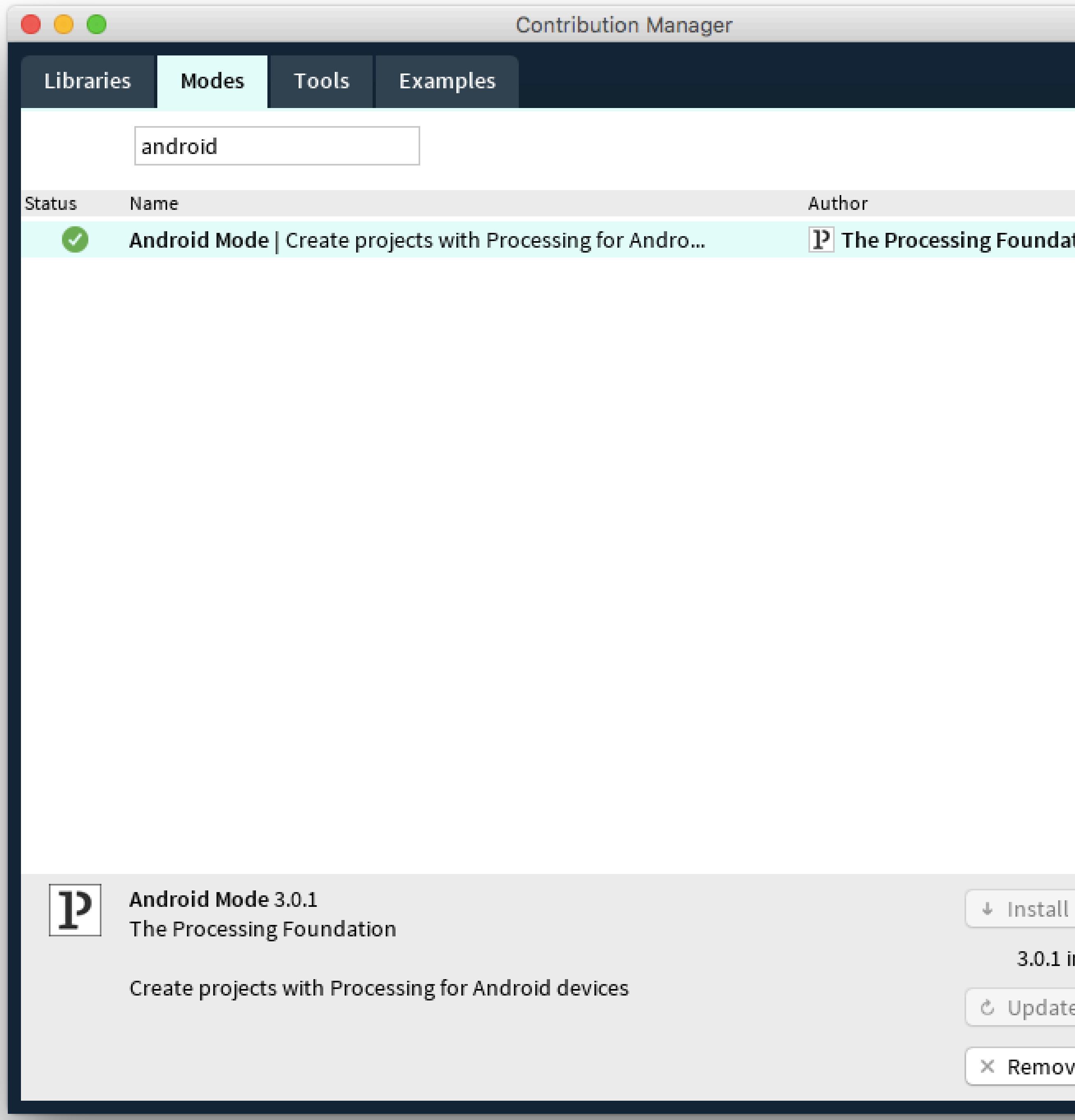
Installation

Before getting started on Processing for Android, make sure you have Processing installed.

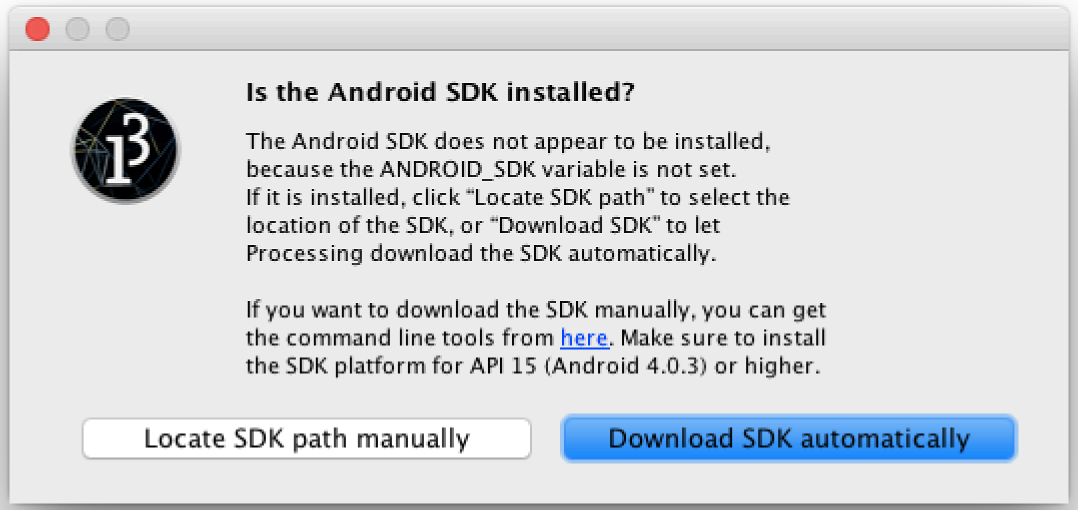
1. Open Processing
2. Click on Add Mode... located in the top right of the window:



3. Search for "android" in the Contribution Manager and choose the option that has the author "The Processing Foundation" and click Install


Contribution Manager
Libraries
Modes
Tools
Examples
android
Status
Name
Author
Android Mode | Create projects with Processing for Andro...
The Processing Foundat
Android Mode 3.0.1
The Processing Foundation
Create projects with Processing for Android devices
Install
3.0.1 i
Update
Remov

4. Once the installation has been complete, switch the mode to Android
5. This window below should pop up. If you do not have the Android SDK, click on "Download SDK automatically" to install it. If you have already installed the Android SDK, then click on "Locate SDK path manually".


Is the Android SDK installed?

The Android SDK does not appear to be installed, because the ANDROID_SDK variable is not set.
If it is installed, click "Locate SDK path" to select the location of the SDK, or "Download SDK" to let Processing download the SDK automatically.

If you want to download the SDK manually, you can get the command line tools from here. Make sure to install the SDK platform for API 15 (Android 4.0.3) or higher.

5,5. For me, I located my SDK path at `/Users/my-username/Library/Android/sdk`

6. You're done!

Read Getting started with Processing for Android online: https://riptutorial.com/processing/topic/7586/getting-started-with-processing-for-android

Chapter 6: Using Processing with Alternative Code Editors like Sublime and Atom

Examples

Using Processing with Sublime Text

To add a language to Sublime Text, you use Package Control. You can do this by pasting the correct Python code (available on the Package Control site linked above), or by downloading the .sublime-package file (also available for download on the site). Once you set up Package Control, restart Sublime Text.

To install the Processing package from Package Control, follow the following steps:

- Execute Tools | Install processing-java in order to install the processing-java tool. This tool is needed to build Processing sketch in command line and editors such as Sublime Text and Atom.
- Open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P)
- Search for 'Package Control: Install Package'
- Search for 'Processing' and install the package
- Restart Sublime Text

After following these steps, you should be able to select Processing as a language. This will facilitate coding in Processing in Sublime Text.

Using Processing with Atom editor

There are several packages which can run Processing sketches in the Atom editor. These instructions use the Script package. There are also available packages for syntax highlighting and autocomplete, which are required for Script to identify Processing filetypes.

Install the Script plugin. Either by running `apm install script` from the command line or search for the package 'script' by *rgbkrk* in the `Install` tab in Atom settings (shortcut `command + ,` or `ctrl + ,`).

Once Script is installed, you will need to install `processing-java`. This tool comes with the main Processing software and is needed to build Processing sketches on the command line and in editors:

- *MacOS:* Run `Tools > Install "processing-java".`

- *Linux:* Add the Processing directory to your `PATH` environment variable (replace `/path/to/processing` with the path where Processing is installed):

  ```
  sudo ln -s /path/to/processing/processing-java /usr/local/bin/
  ```

- *Windows:* Add the Processing directory to your `PATH` environment variable:
    - Open Advanced System Settings either by running `sysdm.cpl` or searching in Control Panel.
    - Click the Environment Variable button on the Advanced tab.
    - Edit the `PATH` variable to include the Processing directory in either the User variables (for just your account) or System variables (for all users).

Now, you can run Processing sketches by running `Packages > Script > Run Script`. The default shortcut is `command + shift + b` or `ctrl + shift + b`, but to further decrease the pain of transition, you can bind the Processing IDE shortcut to run the sketch. In order to do that:

- Open up the Atom Keymap file by running `File > Keymap`
- Paste following lines at the end of the file (feel free to change the binding to whatever you want).

```
'atom-text-editor':
  'ctrl-r': 'script:run'
```

Using Processing with Eclipse

To use Processing in Eclipse, start by creating a new Java project. Then, select `File > Import` and then choose `General > File System` to locate the `core.jar` file. It can be found in `PATH_TO_PROCESSING/core/library/` for Windows or `/Applications/Processing 3.app/Contents/Java/core/library/` for Mac. Once this is completed, right-click on `core.jar` and add it to the build path.

The boilerplate for using Processing in Eclipse is as follows:

```
import processing.core.PApplet;

public class UsingProcessing extends PApplet {

    // The argument passed to main must match the class name
    public static void main(String[] args) {
        PApplet.main("UsingProcessing");
    }

    // method used only for setting the size of the window
    public void settings(){

    }

    // identical use to setup in Processing IDE except for size()
    public void setup(){

    }

    // identical use to draw in Prcessing IDE
    public void draw(){

    }
}
```

The `settings()` method is used to set the size of the window. For example, to create a 400x400 window, write the following:

```
 public void settings(){
    size(400,400);
 }
```

Everything else as outlined in the Hello World documentation in terms of the use of `setup()` and `draw()` applies here.

As a final example, here is the code from the Drawing a Line example were it to be written in Eclipse:

```
import processing.core.PApplet;

public class UsingProcessing extends PApplet {

    // The argument passed to main must match the class name
    public static void main(String[] args) {
        PApplet.main("UsingProcessing");
    }

    // method for setting the size of the window
    public void settings(){
        size(500, 500);
    }

    // identical use to setup in Processing IDE except for size()
    public void setup(){
        background(0);
        stroke(255);
        strokeWeight(10);
    }

    // identical use to draw in Prcessing IDE
    public void draw(){
        line(0, 0, 500, 500);
    }
}
```

Read Using Processing with Alternative Code Editors like Sublime and Atom online: https://riptutorial.com/processing/topic/5594/using-processing-with-alternative-code-editors-like-sublime-and-atom

Chapter 7: Using Processing's P3D renderer

Remarks

A reference for using P3D is found at https://processing.org/tutorials/p3d/ in a tutorial by Daniel Shiffman.

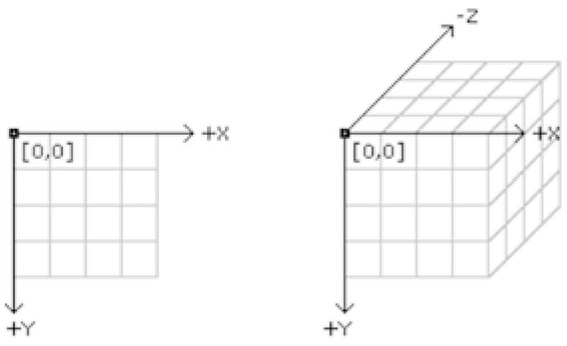
Examples

Getting Started

To be able to use the P3D renderer you must specify it in the `size()` function

```
size(200, 200, P3D);
```

Now, there are three axes: `x`, `y` and `z`.



Now you can proceed in drawing in 3D.

Read Using Processing's P3D renderer online: https://riptutorial.com/processing/topic/7589/using-processing-s-p3d-renderer

Credits

S. No	Chapters	Contributors
1	Getting started with processing	aspiring_sarge, Community, Cows quack, Kevin Workman, MasterBob, Michael Andersen, user2314737
2	Basic shapes and functions using P3D	Cows quack
3	Colours in Processing	Cows quack, MasterBob, Mert Toka
4	Drawing Basic Shapes	Cows quack, MasterBob, sohnryang
5	Getting started with Processing for Android	Cows quack
6	Using Processing with Alternative Code Editors like Sublime and Atom	bakahoe, CodeMacabre, Kevin Workman, Mert Toka, Peter, sohnryang
7	Using Processing's P3D renderer	Cows quack