Planetary Lab for Artists with Processing

José del Carmen Rodríguez Santamaría
The Evoljava Community
http://www.evoljava.com

December 7, 2017

Abstract

The Java Family of programming languages has more than 100 members. One of it is the Processing Language that has been specially targeted to artists. In consequence, it has a very direct path to visual representations and dynamics. We extend here an invitation to learn it and furthermore we show how one can use it to study difficult problems. We illustrate this by exploring some questions in relation to the origin of our planetary system.

Contents

1 INTRODUCTION

Java is perceived by many persons as difficult and by others as unnecessarily difficult. So, a project has arisen whose aim is to capture 80% of its virtues to be reshaped into a new language that could be 10 times easier and directed to visual artists. The result is the **Processing Programming Language**.

The present guide is intended to motivate its study and to explore what it can give of itself with the help of suitable libraries.

2 Starting up

Download and install the package with the environment to develop and run sketches or programs from:

`https://processing.org/`
(All links of this article were verified by December 7/2017.)

Download must be specific for your System. If you do not know what it is, choose Windows 32-bits.

Warning: `Processing` must undergo strong changes to get updated to the new demands of the trade so, future versions can show incompatibilities with the one we used, which is `3.3.6`. In that case, you can download the required version from

`downloads -> stable releases.`

Install the package as usual. Verify that the installation creates a folder with name `sketchbook`. If you want to uninstall `Processing`, you must delete this folder first. After downloading the package with `Processing`, play with installed examples over the menu

`File -> Examples -> Basics -> Arrays.`

Once getting tired, you are ready for an ordered plan that might be the following. Go to

`File -> Examples -> Add examples -> Getting started with Processing -> Install.`

Return to Examples. Open the recently installed folder

`Contributed Examples -> Getting started with Processing.`

Expand the folder of `Chapter 02 Start`. Run the two examples. To run an example, one selects it, opens it, and runs it with the `play icon`.

Always click and drag the mouse over the opening pane because many programs present interactive graphics. By the same token, burn the rest of the

package. If you want to add comprehension to your mind, download the file with the book:

`Getting Started with Processing (2010) by Casey Reas and Ben Fry`
http://cmuems.com/resources/getting_started_with_processing.pdf

Some troubles are expected but they are few indeed and most are related to loading, not existing files. So, try to be not misled by your hurriedness. For instance, `chapter 5 Response` presents programs whose output is not drawn to a pane but printed to the console, which is a black space below the editor pane of Processing, where programs are printed. Other times the program expects you to touch and/or pinch the drawing pane with the mouse. Other times you are expected to press the right button on the mouse or click inside a circle. Or you must strike a key after pinching the drawing pane. How to know what one must do? Read the program with understanding. There are moreover occasions when a difficult technique is introduced in various steps but only the last one produces an output. Chapter 13 needs various libraries, special settings of the System and electronic crafts. Anyway, some programs can be run. To install a library, say, Sound, go over the menus:

`Sketch -> Import Library -> Add library -> Sound -> Install`

`Sound` is expected to run directly over `Intel + Windows`. Else, you need special and mysterious settings of the System. Chapter `Robots` contain programs that can be run.

If it happens to you that there is no program in the editor window to run, most probably you are invited to solve an exercise that you shall find in the textbook else invent one on your own.

To continue, one might work the examples provided by Casey Reas and Ben Fry in their book

`Processing: a programming handbook for visual designers and artists`
https://processing.org/img/learning/Processing-Sample-070607.pdf

The examples are in the package `Processing Handbook second edition`. `Example 7 of Chapter 2 Using Processing` does not run because it has an extra invisible symbol at the very end. So, pose the cursor after the last key which is underscored with a red serpent. Use back-delete to erase that symbol. The key shall remain but the red serpent shall disappear. The same problem might happen variously. Other errors have a pedagogical aim, are detected by the editor and are easy to correct.

The examples provided by Daniel Shiffman in the package `The Nature of Code` teaches us how a natural scientist can express him or herself in the Processing language:

http://natureofcode.com/book/

His examples are backed by a series of videos:

https://vimeo.com/channels/natureofcode

A general problem with Processing is that it is an intelligent language and so, it does too many tasks behind the scenes. The student might understand that programming is magic and so, he will be disappointed when he lacks the expected magical answer. The solution is to recognize that behind the magic there is a lot of programming, of very detailed programming, that makes the task for you. Therefore, one must grasp what the magic does exactly.

Example: The line of code

```
ellipse(50, 50, 30, 60);
```

is per se a Processing sketch or program and can be run to draw an ellipse. Let us consider now the following program:

```
int y = 10;
int speed = 1;

void draw() {
  background(0);
  ellipse(50, y, 30, 60);
  y = y + speed;
}
```

What does happen when one runs this program? One sees a moving ellipse. This unexpected behavior is encoded by two features: first, the command to draw an ellipse is inside a `draw()` function and second the `y` variable is loosen, it is specified at the start but its final limit is not specified. This means indefinite movement. More pathetic, the movement is produced when an object is presented, erased -or slowly faded away- and a new object is drawn a bit further. Nothing of this is given to be known to the User, who needs just to know the direction and velocity of the movement.

In general, to understand how a program operates, make changes. Say, you can silence a line by inserting // at the start. Or you can change a parameter from 5 to 7 (but not to 70). To open a new file for experimentation is immediate.

To save a program or sketch proceed as usual: for the first time go to the menu

```
File-> Save as.
```

For subsequent times go to the menu

```
File -> Save
```

Processing allows a very interesting mode of compiling and running: it is the `Tweak Mode`. You can see it in the promotional video of Processing or here:

Gal Sasson (Portfolio, 2017)
http://galsasson.com/tweakmode/

To repeat this video inside Processing:

1. Open Processing and select the package `The Nature of Code` inside the Menu `Examples`. Expand its tree and find the file: `Chapter06 agents -> program NOC-6.09-Flocking.`
2. Run the program and observe a flock of birds that arises from the nest.
3. Turn the Tweak mode: go to the menu `sketch -> Tweak`.
4. Run anew the sketch or program and put the emerging window aside that the panel with the program remains visible.
5. Observe the code and pay attention to the fact that some numbers are underscored. Pinch anyone of them and sustain the button down. A double arrow must appear that you can control by moving your mouse to the right else to the left. At the same time observe what happens with the running program. You can do the same with other two required programs that are under tabs shoulder to shoulder with your main sketch. The most impressive variable is

   ```
   sep.mult(1.5);
   ```

   in the sketch `Boid`. This variable controls the degree of cohesion of the flock.

To know more about ongoing Processing projects scan the list in the same site, for example,

`gogoam` : http://galsasson.com/gogoam/

You can download the code from:

https://github.com/galsasson/gogoam

The following library of `The Nature of Code` is very important for our own job:

```
chp05 Physicslibraries -> box2d
```

The following are some sketches of this library that might be interesting for you:

CollisionListeningDeletionExercise
// Basic example of controlling an object with our own motion (by attaching a MouseJoint)
// Also demonstrates how to know which object was hit

CollisionsandControlInterface
// Basic example of controlling a square with our own motion (by attaching a MouseJoint)
// Also demonstrates how to know which object was hit

Exercise 5 10 ApplyForceAttractMouse
// Basic example of falling rectangles with collisions.

Exercise 5 10 ApplyForceSimpleWind
// Basic example of falling rectangles with collisions.

Exercise 5 10 AttractionApplyForce
The problem of asteroids that become a planet.

Exercise 5 6 Bridge with squares

LiquidFunTest modeled on small spheres.

Liquidy: liquid falling over sloppy tables.

NOC 5 4:Polygons = complex 2D asteroids

NOC 5 5 MultipleShapes = complex shapes

NOC 5 6 DistanceJoint = coupled object with a spring.

NOC 5 7 RevoluteJoint = mill

NOC 5 9 CollisionListening

From subPackage `toxiclibs`:

Exercise 5 13 SoftBodySquareAdapted = piece of cloth
Exercise 5 15 Force directed graph: molecule
NOC 5 11 SoftStringPendulum: a marvel.
NOC 5 12 SimpleCluster = Molecule
NOC 5 13 AtractRepel = atom with many electrons.

NOC 6 01 Seek trail = vehicles with a trail.

To illustrate how can one use `Processing` to pursue important and difficult questions, we have prepared a study of questions relating to the origin of our home.

3 The origin of planet Earth

To simulate some events related to the questions of the origin of the Earth, we have used the aforementioned library

`Box2D`

that is a physics engine. See:

```
Open source physics engines
Building believable worlds with open source
M. Jones
Published on July 07, 2011
```

https://www.ibm.com/developerworks/library/os-physicsengines/

Our programs can be downloaded from the accompanying zipped file `NinjaPlanetForProcessing.zip`
So, unzip the file to the `Processing` folder `sketchbook` and then open the file `README` to know the context and order in which sketches must be run.

We consider that our material is suitable to teach how to convert curiosity into questions into programs of research. Our experience shows that one can easily develop many programs of interest but that the posed question is by far more important than our results. This means that we have here another challenge for people that like very difficult problems, ones that never are solved but that always admit new ideas, approaches, and calculations. Now, a word of caution: the Earth is in cosmology an insignificant grain of the dust of stars and so to have or not an explanation for it is not a really important matter. A general, plausible explanation suffices. But here, in the *Evoljava Community*, to explain the Earth is the one and only purpose of cosmology. So, you have a full-fledged explanation of the Earth else you have nothing.

4 CONCLUSION

The programming language Processing has been found to be wonderful. Our pleasure goes with a warm thanksgiving to its creators and to the authors of so many and varied libraries. Just with Box2D we have learned how to make science by direct digital experimentation. Now, in regard to planetary systems, our simulations have allowed us to appreciate how difficult is science and how nice is to make digital science with owns hands. In fact, we have been surprised by all that we have achieved. Thanks to our work, we have now a longing for a more professional approach say, with 3D simulations. Processing has an app for 3D graphics but no physics library for that purpose.

5 Backmatter

Our statement for the use of this an all of our material:
LICENSE: unconditional.

ACKNOWLEDGEMENTS:
GRAMMARLY
https://app.grammarly.com/