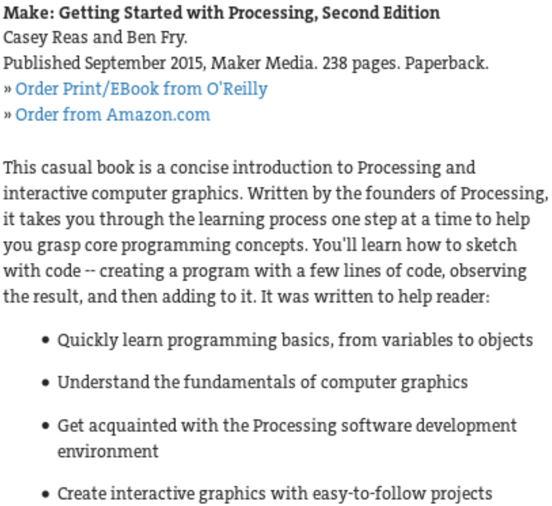
Physics 8 — Wednesday, November 22, 2017

- HW11 due Friday, December 1.
- Today: a tutorial of the "Processing" computer programming language — whose purpose is to learn how to code within the context of the visual arts. It makes coding fun and visual.
- Extra-credit options (if you're interested):
  - Read Onouye Ch 9 (columns) and write 1–2 pages summarizing what you learned.
  - Read Mazur Ch 13 (gravity) and/or Ch 14 (special relativity) and summarize what you learned.
  - Write up your response to podcast about near-fatal flaw in Citigroup Center, 601 Lexington Ave, NYC `http://positron.hep.upenn.edu/p8/files/citicorp_tower.mp3`
  - Learn to use Mathematica (ask me how), which is a system for doing mathematics by computer. (It is the brains behind Wolfram Alpha.) Penn's site license makes Mathematica free-of-charge for SAS and Wharton students.
  - Use "Processing" to write a program to draw or animate something that interests you. (Not necessarily physics-related.)
  - Knowing "how to code" is empowering & enlightening. So I offer you an excuse to give it a try, for extra credit, if you wish.

Second Edition
Make:
Getting Started with Processing
A Hands-On Introduction to Making Interactive Graphics
Casey Reas & Ben Fry



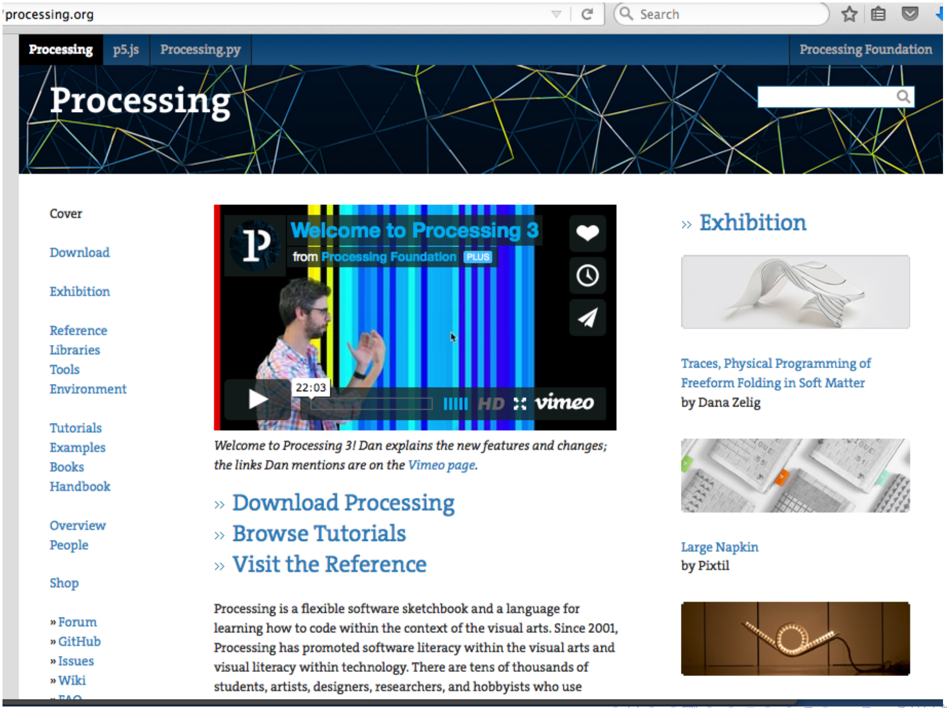
Make: Getting Started with Processing, Second Edition
Casey Reas and Ben Fry.
Published September 2015, Maker Media. 238 pages. Paperback.
» Order Print/EBook from O'Reilly
» Order from Amazon.com

This casual book is a concise introduction to Processing and interactive computer graphics. Written by the founders of Processing, it takes you through the learning process one step at a time to help you grasp core programming concepts. You'll learn how to sketch with code -- creating a program with a few lines of code, observing the result, and then adding to it. It was written to help reader:

- Quickly learn programming basics, from variables to objects
- Understand the fundamentals of computer graphics
- Get acquainted with the Processing software development environment
- Create interactive graphics with easy-to-follow projects

The software is free & open-source. Runs on Mac, Windows, Linux. The "getting started" book will set you back about $15.

or start with the in-browser video tutorial (no download needed):
`http://hello.processing.org`



Cover

Download

Exhibition

Reference
Libraries
Tools
Environment

Tutorials
Examples
Books
Handbook

Overview
People

Shop

» Forum
» GitHub
» Issues
» Wiki

Welcome to Processing 3! Dan explains the new features and changes; the links Dan mentions are on the Vimeo page.

» Download Processing
» Browse Tutorials
» Visit the Reference

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use

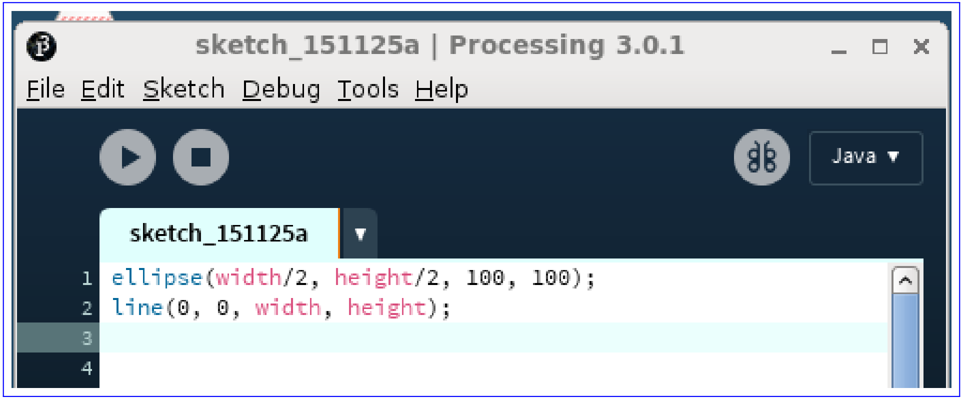
» Exhibition

Traces, Physical Programming of Freeform Folding in Soft Matter
by Dana Zelig

Large Napkin
by Pixtil

“hello world” program

Let’s draw a circle and a line.


sketch_151125a | Processing 3.0.1
File Edit Sketch Debug Tools Help
Java ▼
sketch_151125a

```
ellipse(width/2, height/2, 100, 100);
line(0, 0, width, height);
```

More commonly, a Processing program has a function called `setup()` that runs once when the program starts, and another function called `draw()` that runs once per frame.

```
void setup() {
  // this function runs once when the program starts up
  size(900, 450);  // sets width & height of window (in pixels)
}

void draw() {
  // this function runs once per frame of the animation
  line(0, frameCount, width, height-frameCount);
}
```

Let's make it do something repetitive

```
void setup() {
  // this function runs once when the program starts up
  size(900, 450);  // sets width & height of window (in pixels)
}

void draw() {
  // this function runs once per frame of the animation
  float dy = 0.5*height + 0.5*height*sin(0.01*frameCount);
  line(0, dy, width, height-dy);
}
```

How about repeating something more exciting?

```
void setup() {
  // this function runs once when the program starts up
  size(900, 450);  // sets width & height of window (in pixels)
}

void draw() {
  // this function runs once per frame of the animation
  float dy = 0.5*height + 0.5*height*sin(0.01*frameCount);
  line(0, dy, width, height-dy);
  float t = 0.02*frameCount;
  float x = 0.5*width + 200*cos(t);
  float y = 0.5*height + 200*sin(t);
  ellipse(x, y, 20, 20);
}
```

Did you ever have a Spirograph toy when you were a kid?

```
void setup() {
  size(900, 450);
}

void draw() {
  float t = 0.02*frameCount;
  float x = 0.5*width + 200*cos(t) + 30*cos(11*t);
  float y = 0.5*height + 200*sin(t) - 30*sin(11*t);
  ellipse(x, y, 5, 5);
}
```

How about something that starts to resemble physics? A really, really low-tech animation of an planet orbiting a star.

```
void setup() {
  size(900, 450);
}

void draw() {
  float t = 0.01*frameCount;
  float xsun = 0.5*width;
  float ysun = 0.5*height;
  ellipse(xsun, ysun, 20, 20);
  float rplanet = 200;
  float xplanet = xsun + rplanet*cos(t);
  float yplanet = ysun + rplanet*sin(t);
  ellipse(xplanet, yplanet, 10, 10);
}
```

Let's add a moon in orbit around the planet.

```
void draw() {
  float t = 0.01*frameCount;
  float xsun = 0.5*width;
  float ysun = 0.5*height;
  // clear screen before each new frame
  background(128);
  // draw sun
  ellipse(xsun, ysun, 20, 20);
  float rplanet = 200;
  float xplanet = xsun + rplanet*cos(t);
  float yplanet = ysun + rplanet*sin(t);
  // draw planet
  ellipse(xplanet, yplanet, 10, 10);
  float rmoon = 30;
  float xmoon = xplanet + rmoon*cos(t*365/27.3);
  float ymoon = yplanet + rmoon*sin(t*365/27.3);
  // draw moon
  ellipse(xmoon, ymoon, 5, 5);
}
```

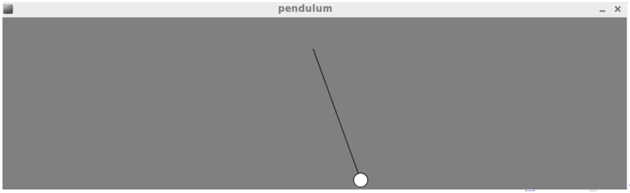
How about adding an inner planet?

```
void draw() {
  ... other stuff suppressed ...
  // draw moon
  ellipse(xmoon, ymoon, 5, 5);
  // add second planet
  float year_mercury_days = 115.88;  // from Wikipedia
  float T_ratio = year_mercury_days/365.25;
  float R_ratio = pow(T_ratio, 0.6667);
  xplanet = xsun + R_ratio*rplanet*cos(t/T_ratio);
  yplanet = ysun + R_ratio*rplanet*sin(t/T_ratio);
  ellipse(xplanet, yplanet, 7, 7);
}
```

Animate a pendulum

```
void draw() {
  float t = 0.01*frameCount;
  float g = 9.8;
  float L = 2.0;
  float degree = PI/180.0;
  float amplitude = 20*degree;
  float omega = sqrt(g/L);
  float theta = amplitude * sin(omega*t);
  float xbob = L * sin(theta);
  float ybob = L * cos(theta);
  // convert coordinates into pixel coordinates
  ... continued on next slide ...
}
```

```
void draw() {
  ... continued from previous slide ...
  // convert coordinates into pixel coordinates
  float xpixel_pivot = 0.5*width;
  float ypixel_pivot = 0.1*height;
  float scale = 100.0;  // pixels per meter
  float xpixel_bob = xpixel_pivot + scale*xbob;
  float ypixel_bob = ypixel_pivot + scale*ybob;
  // clear the screen for each new frame of animation
  background(128);
  // draw the string
  line(xpixel_pivot, ypixel_pivot, xpixel_bob, ypixel_bob);
  // draw the bob
  ellipse(xpixel_bob, ypixel_bob, 20, 20);
}
```

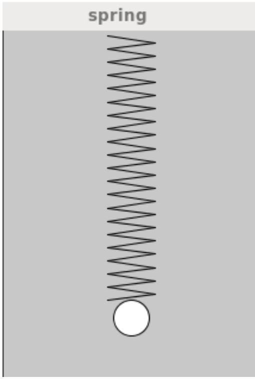


Animate a mass bobbing on a spring

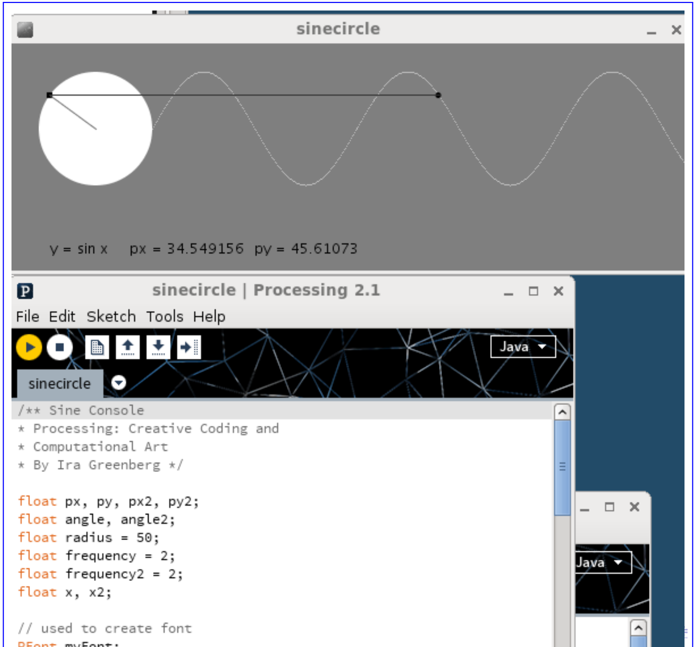
```
void draw() {
  float t = 0.01*frameCount;
  float omega = 1.0;
  float amplitude = 0.5;
  float Lequilibrium = 2.0;
  float xbob = 0;
  float ybob = Lequilibrium + amplitude * cos(omega*t);
  float xpixel_anchor = 0.5*width;
  float ypixel_anchor = 0.01*height;
  float scale = 100.0;
  float xpixel_bob = xpixel_anchor + scale*xbob;
  float ypixel_bob = ypixel_anchor + scale*ybob;
  // draw the bob
  float rbob = 15;
  ellipse(xpixel_bob, ypixel_bob, 2*rbob, 2*rbob);
}
```

Clear screen between frames; draw the spring

```
void draw() {
  ... other stuff suppressed ...
  // clear the screen for each new frame
  background(200);
  // draw the bob
  float rbob = 15;
  ellipse(xpixel_bob, ypixel_bob, 2*rbob, 2*rbob);
  // draw the spring as a series of zig-zag lines
  int nzigzag = 20;
  for (int i=0; i<nzigzag; i++) {
    float spring_top = ypixel_anchor;
    float spring_bottom = ypixel_bob - rbob;
    float dy = (spring_bottom-spring_top)/nzigzag;
    float xzig = xpixel_anchor - 20;
    float yzig = ypixel_anchor + i*dy;
    float xzag = xpixel_anchor + 20;
    float ymid = yzig + 0.5*dy;
    float yzag = yzig + dy;
    line(xzig, yzig, xzag, ymid);
    line(xzag, ymid, xzig, yzag);
  }
```



Here's a more elaborate example, from the online Processing tutorials pages: http://processing.org/tutorials/trig/



```
y = sin x    px = 34.549156  py = 45.61073
```

```
/** Sine Console
 * Processing: Creative Coding and
 * Computational Art
 * By Ira Greenberg */

float px, py, px2, py2;
float angle, angle2;
float radius = 50;
float frequency = 2;
float frequency2 = 2;
float x, x2;

// used to create font
PFont myFont;
```

Back to the spring: let's add some "physics" to it.

```
// we will update position & velocity frame-by-frame
float y = 1.49;
float vy = 0.0;

void draw() {
  float dt = 0.01;
  float k = 20.0;
  float m = 1.0;
  float g = 9.8;
  float Lrelaxed = 1.0;
  float omega = sqrt(k/m);
  y = y + vy*dt;
  float Fy = m*g - k*(y-Lrelaxed);
  vy = vy + (Fy/m)*dt;
  float xbob = 0;
  float ybob = Lrelaxed + y;
  ... the rest is unchanged ...
```

https://en.wikipedia.org/wiki/Leapfrog_integration

- If this looks interesting to you, then I recommend that you start with this easy online video tutorial that will help you get started with coding in Processing in about an hour! No download or software install is needed for this tutorial — you type your first programs directly into your web browser as you follow along with the video.
  `http://hello.processing.org`
- I tried to animate a bending beam, but I failed utterly!
- If you're in Addams Hall often, you might ask Orkan Telhan if he has ideas — I believe he still teaches Processing in FNAR 264 / VLST 264, "Art, Design, and Digital Culture."
- There are also tons of examples at `http://processing.org` that you could use as starting points or for inspiration.

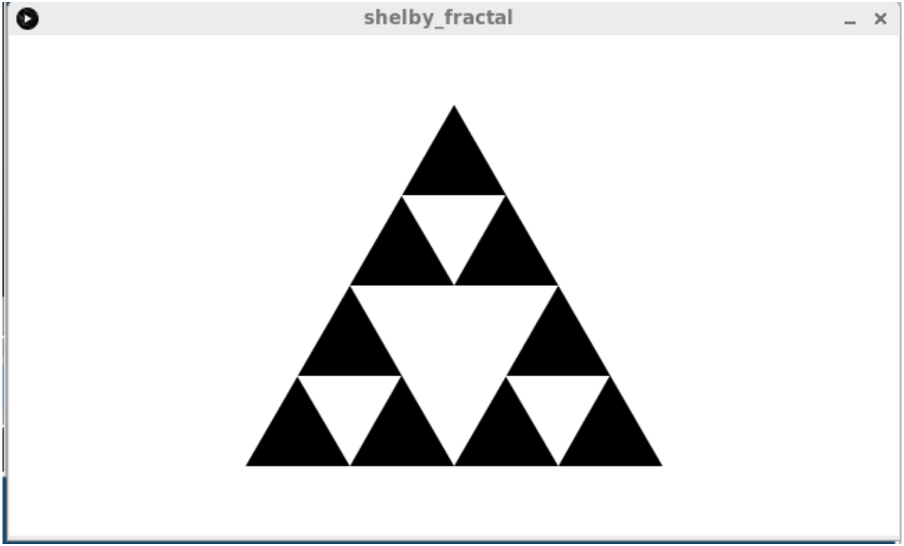
- If you're feeling super-ambitious, and you want to do something highly physics-oriented (which might require you to read Mazur's Chapter 13 on Gravity, which would earn you additional extra-credit), then you could start with my "dumbplanets" example (Sun, Mercury, Earth, Moon) and convert it to use "physics," i.e. to evaluate the equations of motion for each time step.
- You would need to learn (unless you know it from high-school physics) about Newton's universal gravitational force:

$$F = \frac{Gm_1 m_2}{r^2}$$

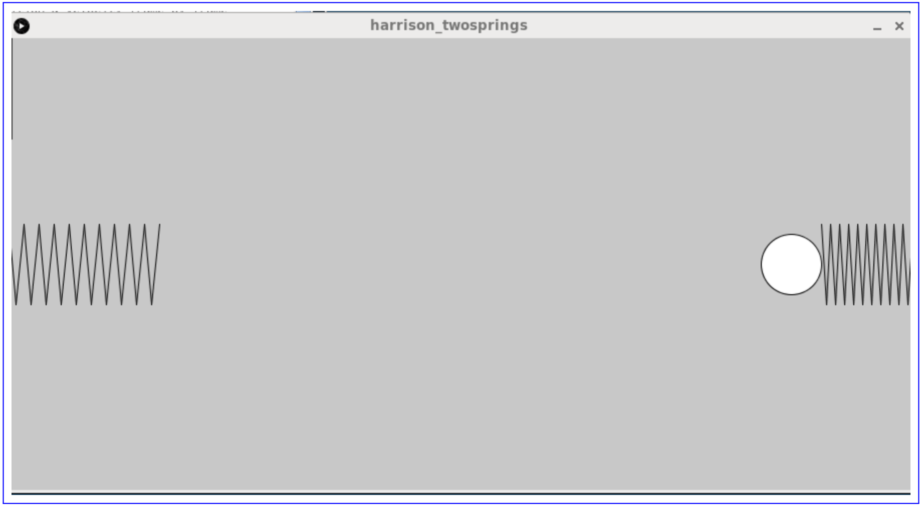
- This would be a simplification of a CIS110 project idea: `www.cis.upenn.edu/~cis110/12fa/hw/hw02/nbody.shtml`

- But you're welcome to do something much simpler, and not necessarily at all physics-related!

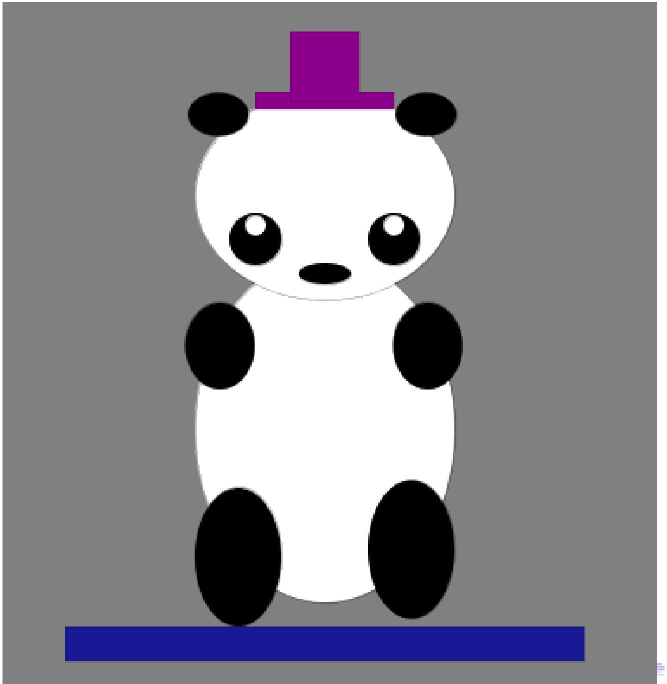
An example from a Fall 2013 student: drawing a fractal.



Another Fall 2013 student: ball bouncing between two springs

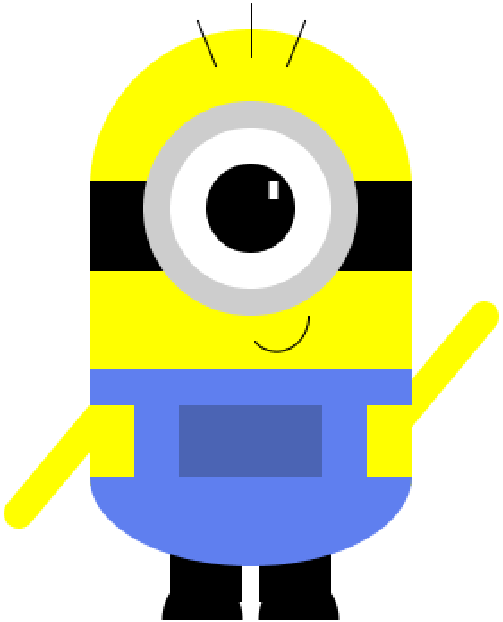


An example from a Fall 2015 student: an animated panda.



An example from a Fall 2015 student: a rotating fractal.

An example from a Fall 2015 student: a minion.



Fall 2015 student: bird moves where you move the mouse pointer.

Physics 8 — Wednesday, November 22, 2017

- HW11 due Friday, December 1.
- Today: a tutorial of the "Processing" computer programming language — whose purpose is to learn how to code within the context of the visual arts. It makes coding fun and visual.
- Extra-credit options (if you're interested):
  - Read Onouye Ch 9 (columns) and write 1–2 pages summarizing what you learned.
  - Read Mazur Ch 13 (gravity) and/or Ch 14 (special relativity) and summarize what you learned.
  - Write up your response to podcast about near-fatal flaw in Citigroup Center, 601 Lexington Ave, NYC `http://positron.hep.upenn.edu/p8/files/citicorp_tower.mp3`
  - Learn to use Mathematica (ask me how), which is a system for doing mathematics by computer. (It is the brains behind Wolfram Alpha.) Penn's site license makes Mathematica free-of-charge for SAS and Wharton students.
  - Use "Processing" to write a program to draw or animate something that interests you. (Not necessarily physics-related.)
  - Knowing "how to code" is empowering & enlightening. So I offer you an excuse to give it a try, for extra credit, if you wish.