Processing

http://processing.org

Presentation by
Ben Leduc-Mills

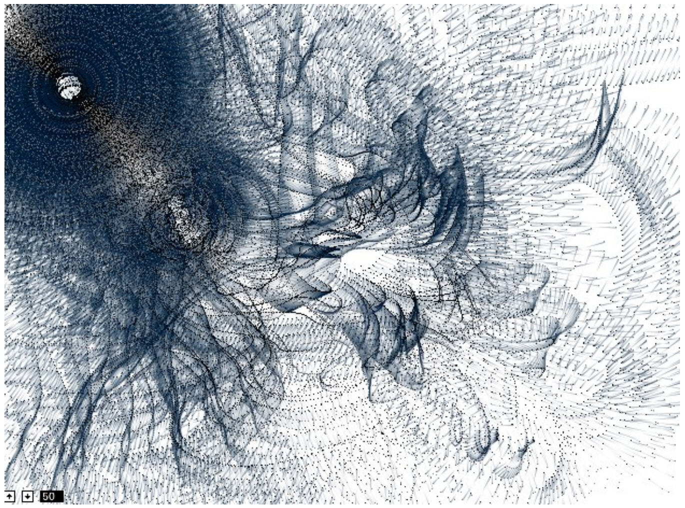
Processing: History

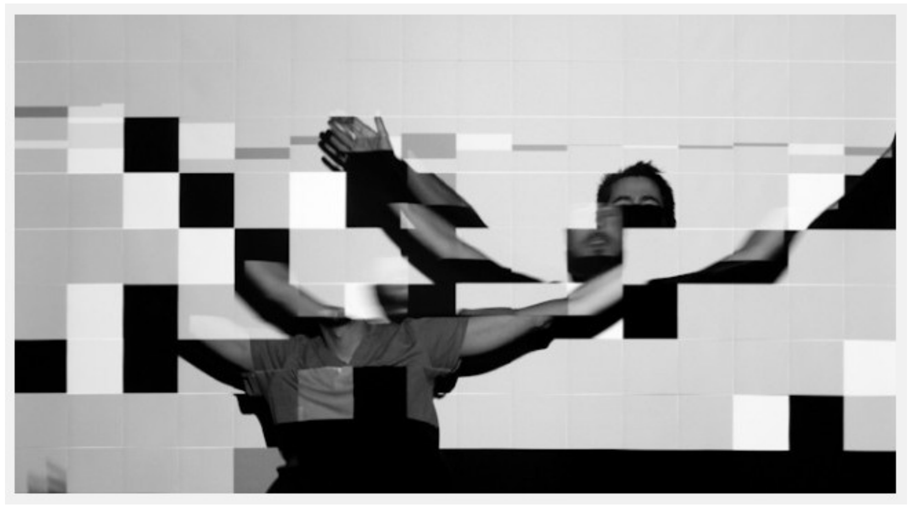
Processing is a free, open-source Java-based framework as well as an Integrated Development Environment (IDE).

It was initially developed by Casey Reas and Ben Fry at the MIT Media Lab in 2001 as a tool to help teach programming, with special attention given to artistic and visual applications.

Processing has since gone on to become one of the most widely used tools for teaching introductory programming to non-computer scientists, as well as a popular tool amongst artists and creative technologists for realizing their work.

Processing: Examples





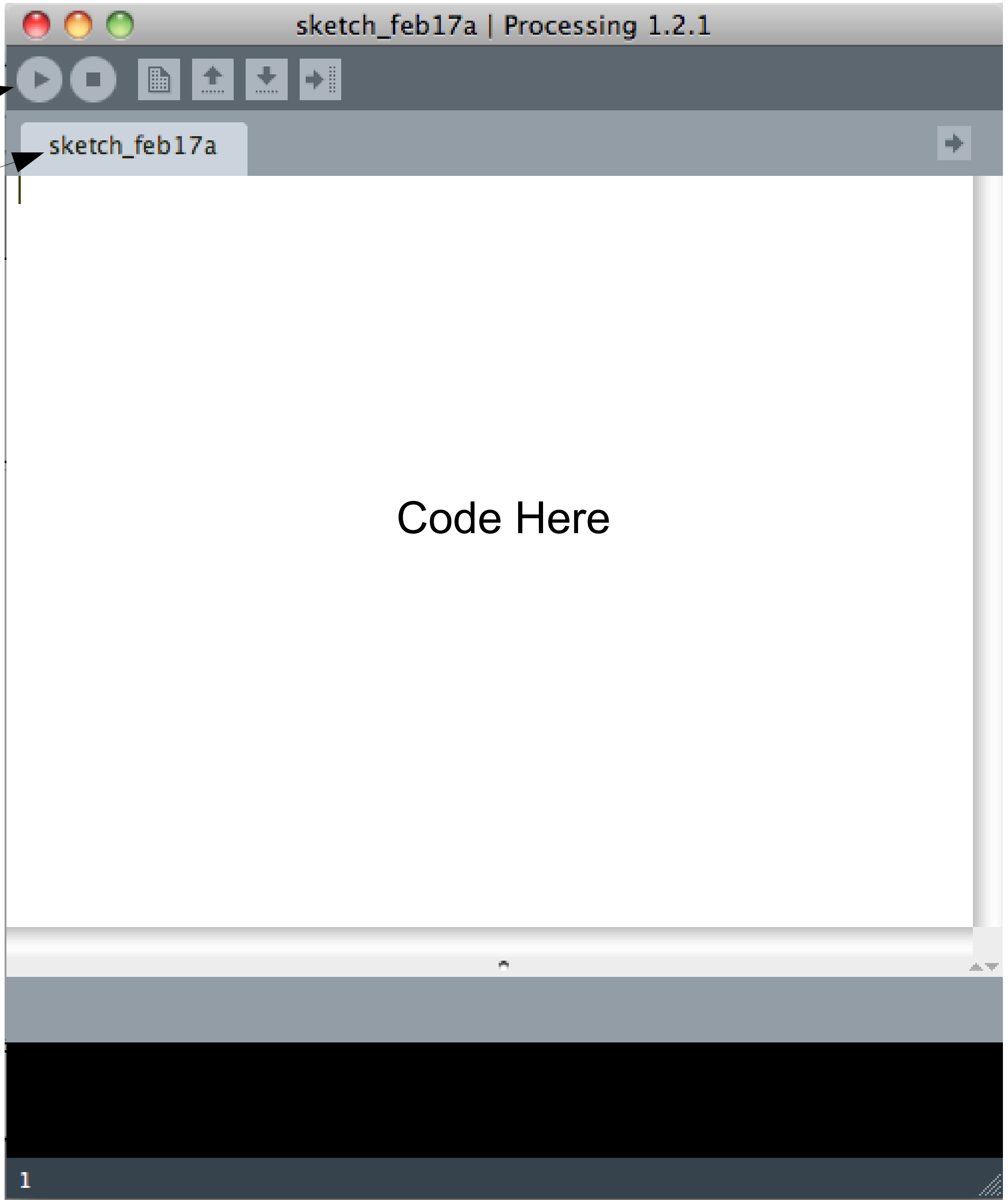
Things Processing is Good At:

- Data Visualization
- Algorithmic Drawing / Generative Visualization
- Computer Vision
- Real-time visual processing
- Hardware Interfacing (serial communication)
- Physics / Math Simulations
- Pretty Graphics in General

Processing: The IDE (I)

Program controls
(see next slide)

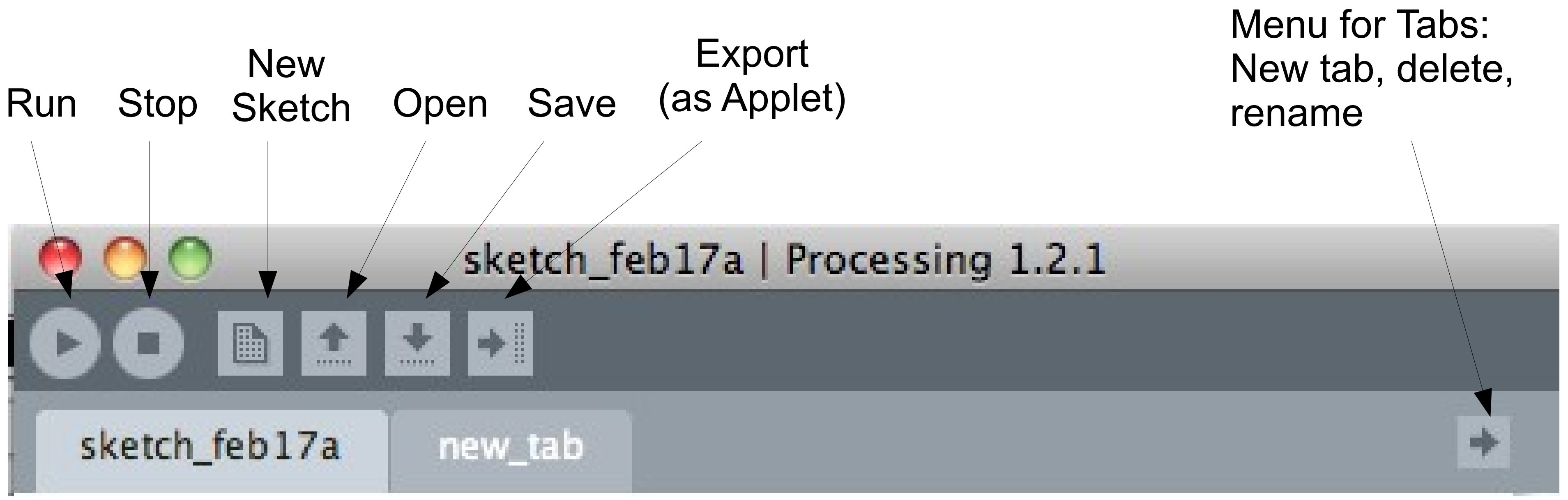
Programs are
Called 'sketches'



Debug/
Console Area

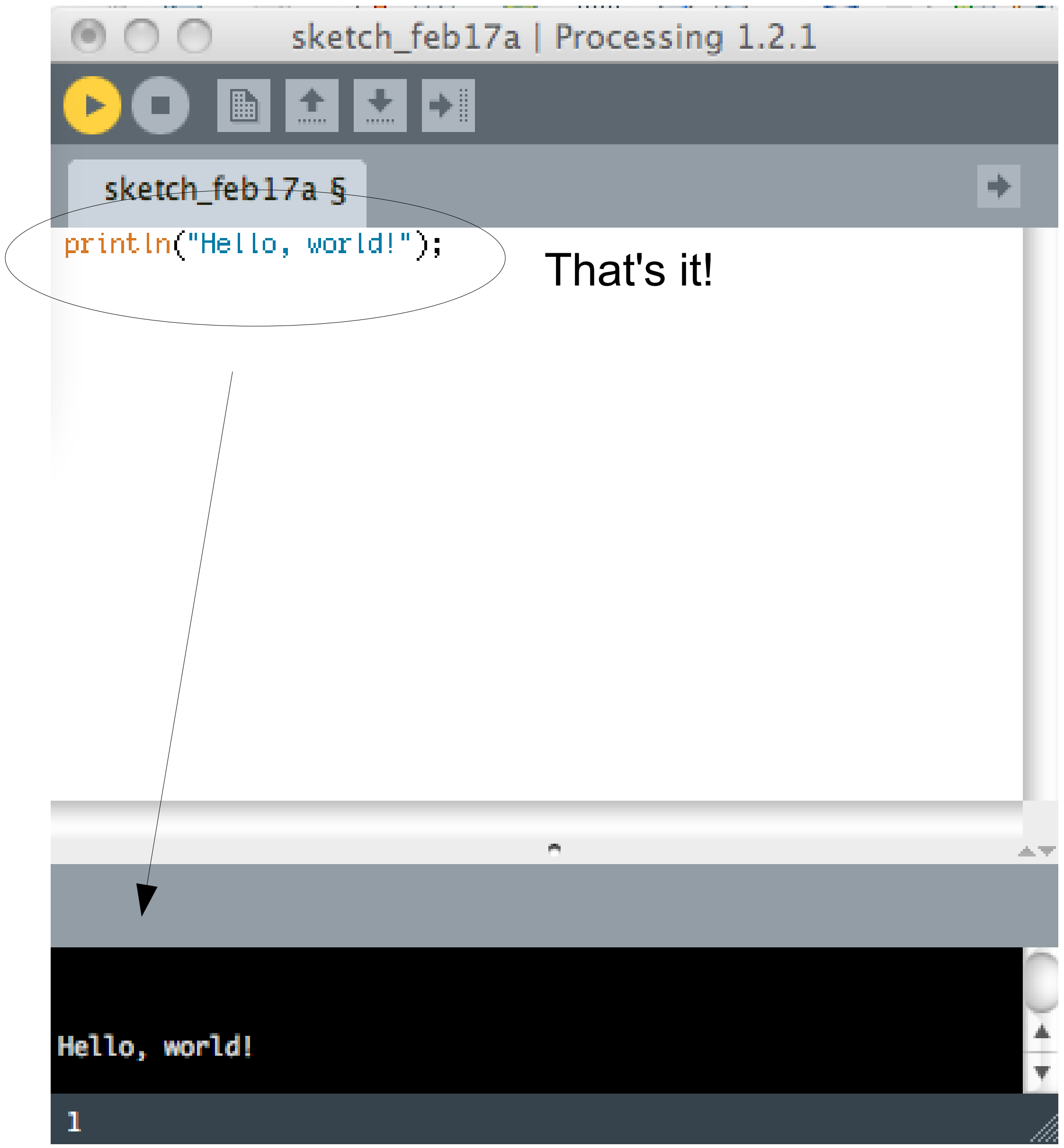
Line Number

Processing: The IDE (II)


Run
Stop
New Sketch
Open
Save
Export (as Applet)
Menu for Tabs: New tab, delete, rename
sketch_feb17a | Processing 1.2.1
sketch_feb17a
new_tab

Note: Tabs are not necessarily separate classes in a package, but it's a useful way to think of them - all tabs belong to the same sketch and can reference each other's methods and object instances.

Processing: Hello World (I)



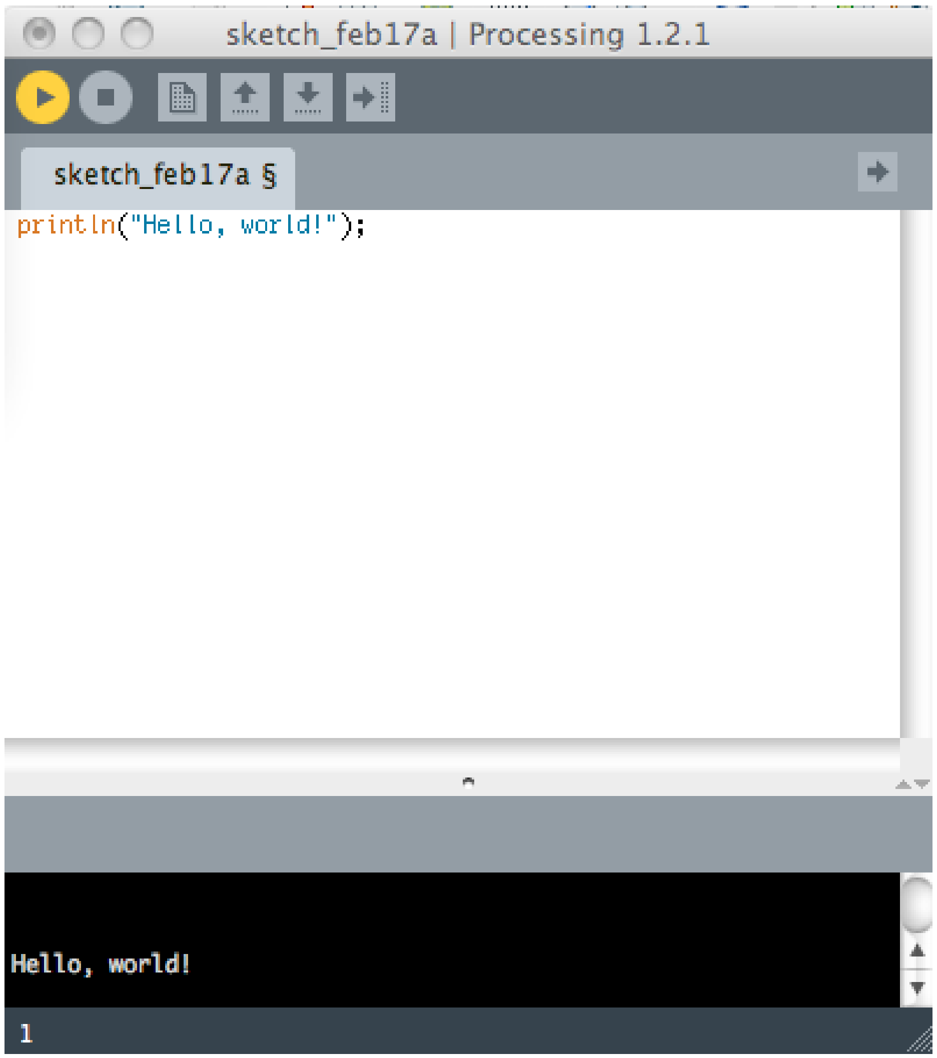
```
println("Hello, world!");
```

That's it!

```
Hello, world!
```

Processing: Hello World (II)

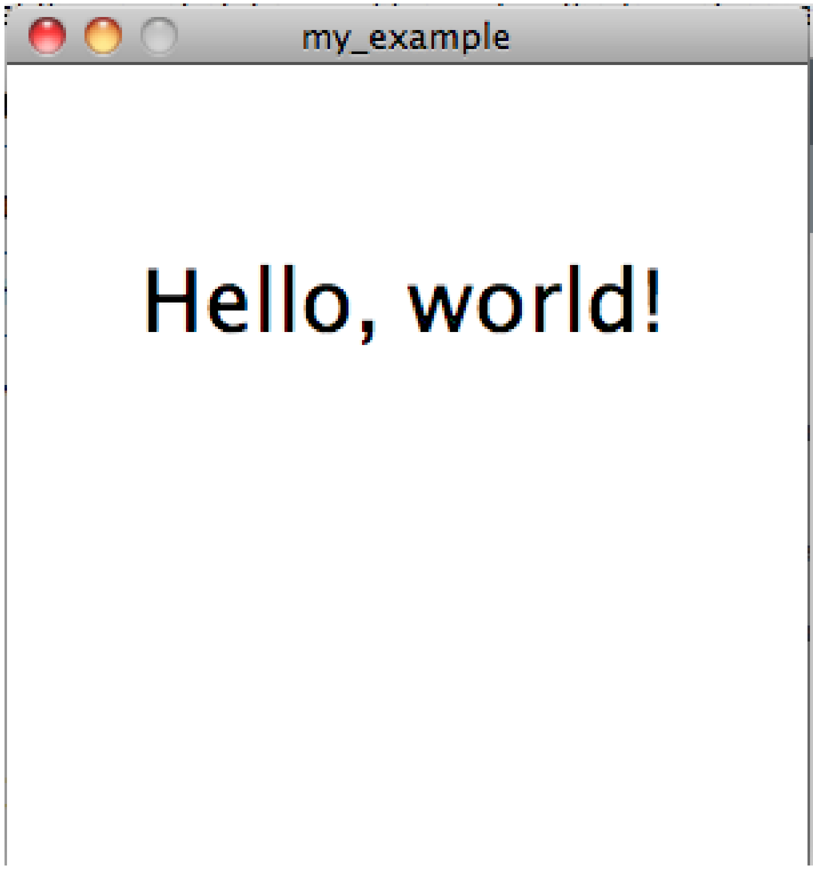
- Each processing sketch is actually a subclass of a PApplet superclass which defines most of Processing's behavior.
- i.e., Processing encapsulates much of the intricacies of developing in Java (don't need a main method, public/private distinctions)
- In fact, static methods are prohibited, unless you're programming in pure Java mode (more on this later)

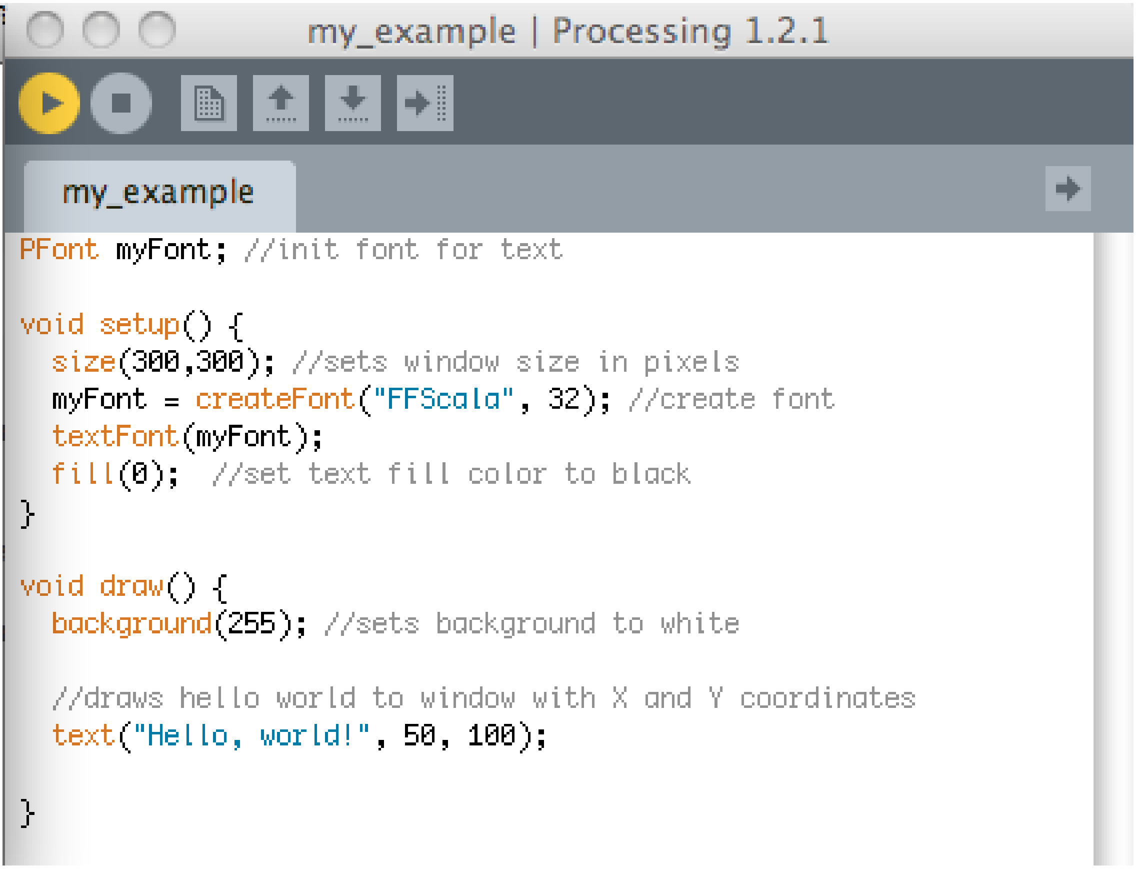


```
println("Hello, world!");
```

```
Hello, world!
```

Processing: A Better Example (I)





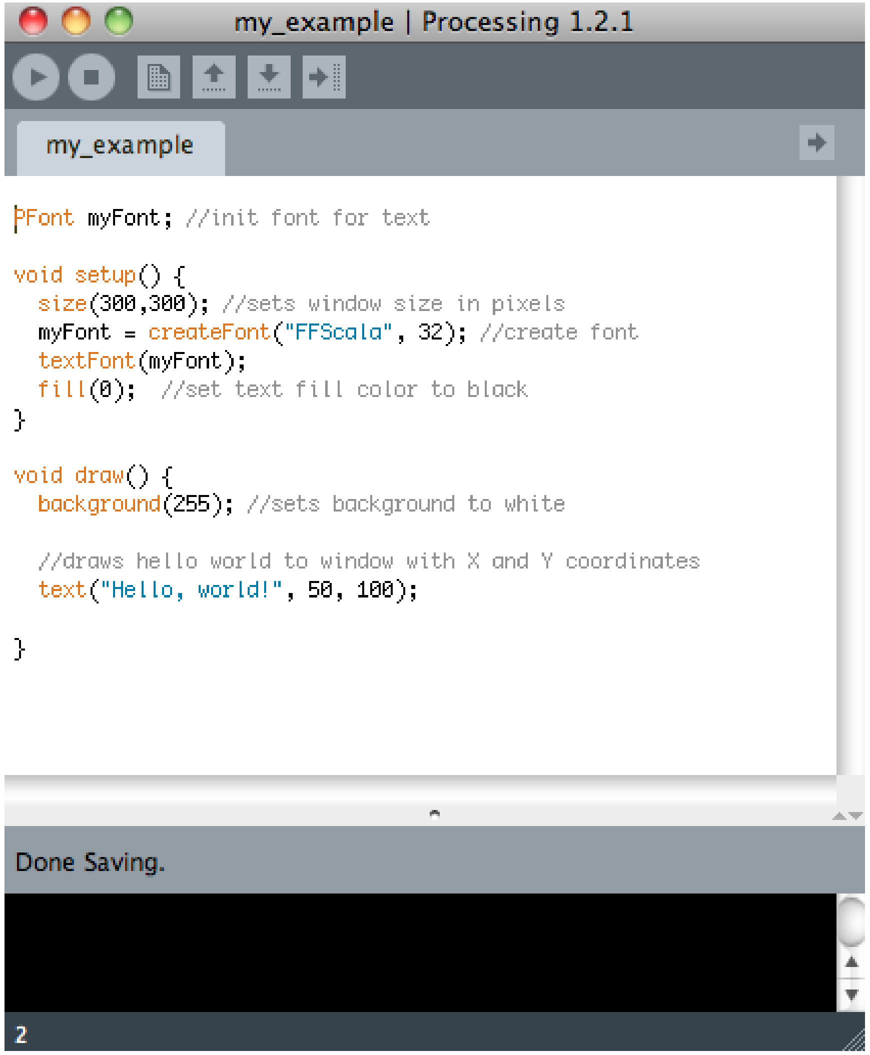
```
PFont myFont; //init font for text

void setup() {
  size(300,300); //sets window size in pixels
  myFont = createFont("FFScala", 32); //create font
  textFont(myFont);
  fill(0);  //set text fill color to black
}

void draw() {
  background(255); //sets background to white

  //draws hello world to window with X and Y coordinates
  text("Hello, world!", 50, 100);

}
```

Result

Code

Processing: A Better Example (II)

- The setup() and draw() methods are common to most Processing sketches
- Setup() is for setup - initializing variables and objects. Only gets run once on startup.
- Draw() is the main loop of your sketch – it will continue to loop unless a noLoop() method is called.
- Basically, if it changes (like an animation), it needs to be called in the draw() method.



```
PFont myFont; //init font for text

void setup() {
  size(300,300); //sets window size in pixels
  myFont = createFont("FFScala", 32); //create font
  textFont(myFont);
  fill(0);  //set text fill color to black
}

void draw() {
  background(255); //sets background to white

  //draws hello world to window with X and Y coordinates
  text("Hello, world!", 50, 100);

}
```

```
Done Saving.
```

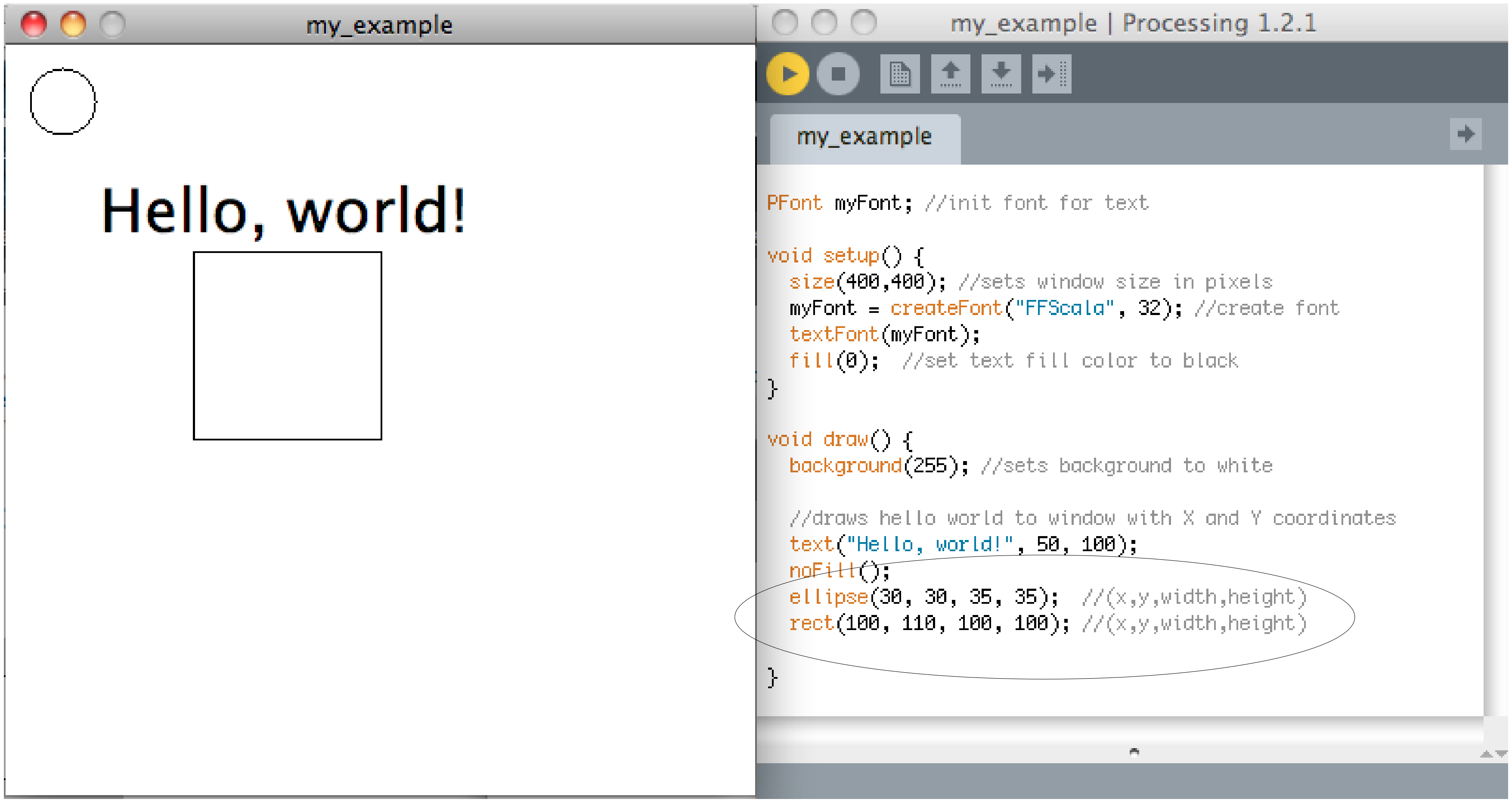
Processing: Built-In Functions (I)

- Processing has a large amount of built-in functionality aimed at simplifying common tasks – it 'gives a lot for free'

  For example, System.out.println() becomes println()
- However, most standard structures, data types, operations, (and operators) are the same (or very similar) as in Java
- The entire basic library reference can be found here: http://processing.org/reference/

Processing: Build-In Functions(II)

There are 2D and 3D primitives that make drawing some shapes very easy: arc(), ellipse(), line(), point(), rect(), box(), sphere(), etc.



```
PFont myFont; //init font for text

void setup() {
  size(400,400); //sets window size in pixels
  myFont = createFont("FFScala", 32); //create font
  textFont(myFont);
  fill(0);  //set text fill color to black
}

void draw() {
  background(255); //sets background to white

  //draws hello world to window with X and Y coordinates
  text("Hello, world!", 50, 100);
  noFill();
  ellipse(30, 30, 35, 35);  //(x,y,width,height)
  rect(100, 110, 100, 100); //(x,y,width,height)

}
```

Processing: Built-In Functions (II)

- Processing also gives very easy access to various types of user input:

  mouseClicked(), mouseDragged(), mouseX(), mouseY(), keyPressed(), keyReleased() are all native functions.

- Processing also has classes and methods for loading and manipulating images (PImage) and text (PFont)

Processing: Standard Libraries

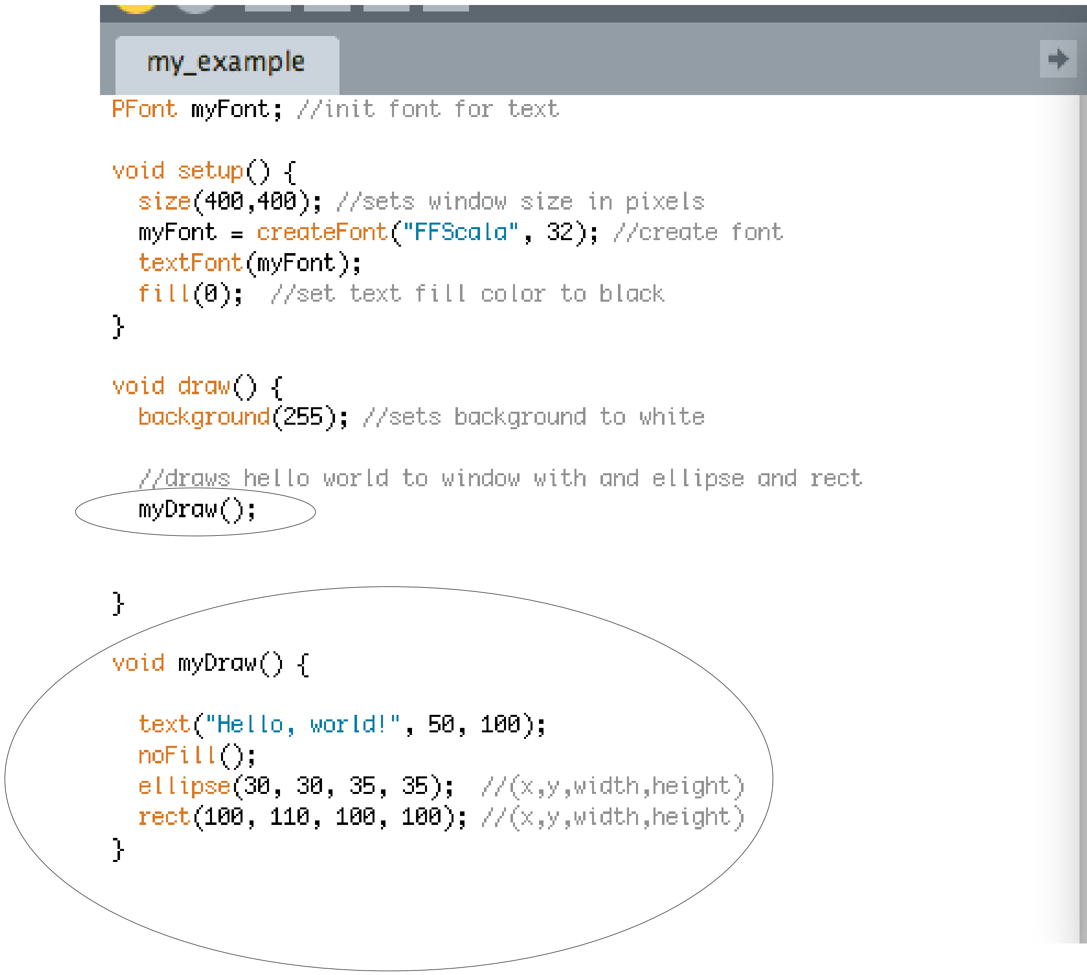
Processing ships with standard libraries that can significantly extend the capabilities of Processing:

- **Video** – interface to QuickTime using a camera, playing, and creating movies
- **Network** – send and receive data by creating clients and servers
- **Serial** – send data via serial between Processing and external hardware (RS-232)
- **PDF Export** – export your sketches in .pdf format
- **OpenGL** – support for OpenGL accelerated sketches – uses JOGL
- **Minim** – easy to use audio library based on the JavaSound API
- **DXF Export** – allow export of 3D shapes to .dxf format
- **Arduino –** allows direct control of an Arduino microcontroller board
- **Javascript** – interface between processing applets and javascript

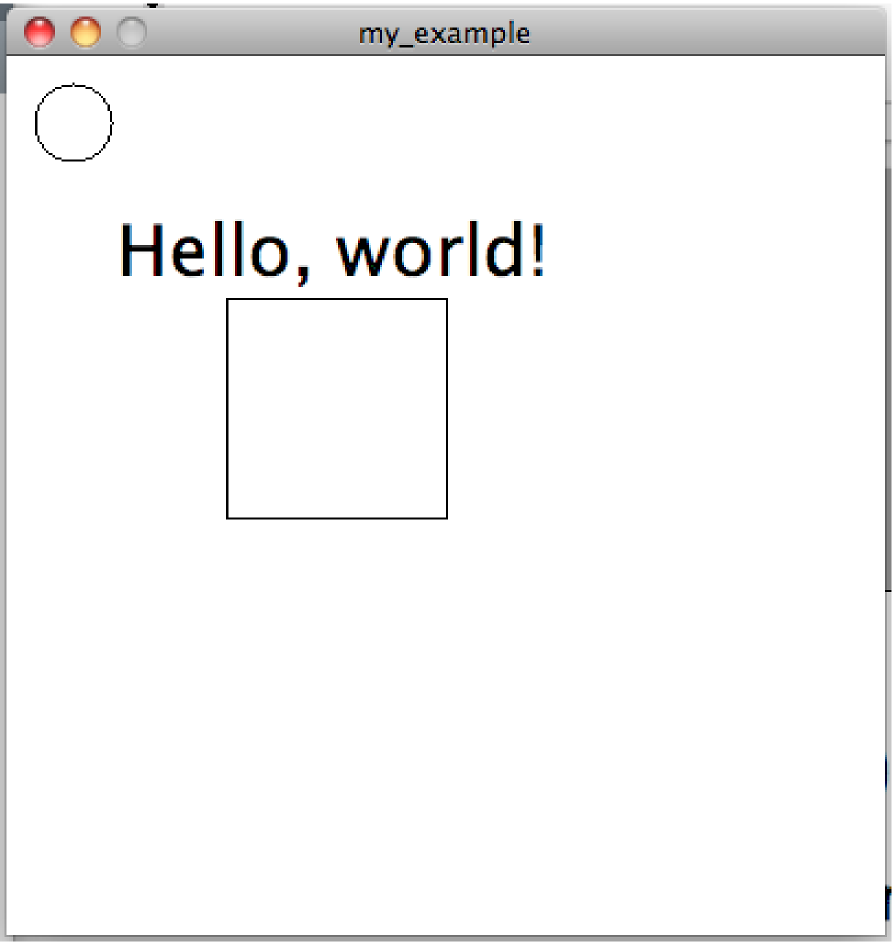
Processing: Contributed Libraries

- In addition to the libraries that ship with Processing, there are numerous (~100) contributed libraries that further extend Processing's capabilities.
- Libraries focus a variety of topics: sound, graphics, geometry, 3D, animation, computer vision, file import/export, hardware interface, and more
- However, they are not always as reliable or well-maintained as the main libraries, so use at your own risk
- For a list of the best contributed libraries, visit: http://processing.org/reference/libraries/

Processing: User-Defined Methods



```
PFont myFont; //init font for text

void setup() {
  size(400,400); //sets window size in pixels
  myFont = createFont("FFScala", 32); //create font
  textFont(myFont);
  fill(0);  //set text fill color to black
}

void draw() {
  background(255); //sets background to white

  //draws hello world to window with and ellipse and rect
  myDraw();


}

void myDraw() {

  text("Hello, world!", 50, 100);
  noFill();
  ellipse(30, 30, 35, 35);  //(x,y,width,height)
  rect(100, 110, 100, 100); //(x,y,width,height)
}
```



Same result as before, but we put the drawing functionality into our own myDraw() function

Processing: User-Defined Classes(I)

Class declaration is similar to Java, but without scope declarations (you can put them in if you like).

Variables are assumed to be private, methods are assumed to be public.

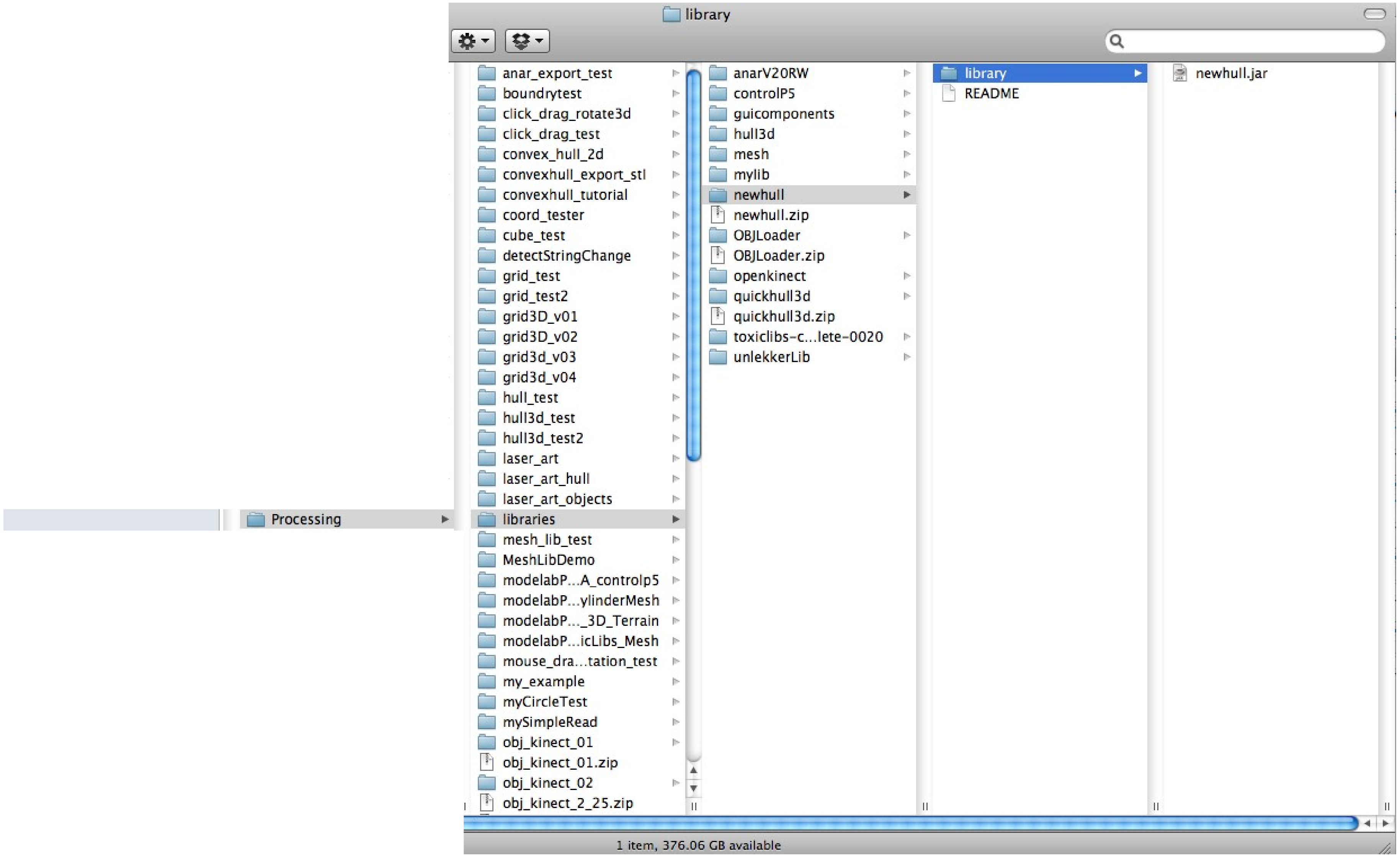
You can use any of Processing's core or contributed library methods within your own class, providing you import everything you need.

```
class MShape {

  int numPoints;
  int xMax, xMin;
  int yMax, yMin;
  int zMax, zMin;
  float strokeW;

  MShape(int inumPoints, int ixMin, int ixMax, int iyMin, int iyMax, int izMin, int izMax, float istrokeW) {

    numPoints = inumPoints;
    xMin = ixMin;
    xMax = ixMax;
    yMin = iyMin;
    yMax = iyMax;
    zMin = izMin;
    zMax = izMax;
    strokeW = istrokeW;
  }

  void display() {

    beginShape();
    strokeWeight(strokeW);
    for( int i = 0; i < numPoints; i++) {

      int randX = int(random(xMin, xMax));
      int randY = int(random(yMin, yMax));
      int randZ = int(random(zMin, zMax));
      vertex(randX, randY, randZ);
    }
    endShape();
  }
}
```

Processing: User-Defined Classes(II)

- You can also make any regular or custom java class or library available to your Processing sketch.
- First, export the class or library as a .jar file
- Put the .jar file in a folder called 'library'
- Put the whole thing in another folder named after your library or class (e.g. MyClass)
- Drag the whole thing to your Processing → Libraries folder, and **restart** Processing
- You should see your library appear under the Sketch → Import Libraries... menu
- Now you can import the library (and use all its methods) in any processing sketch you want

A clearer screenshot:



library

Processing

anar_export_test
boundrytest
click_drag_rotate3d
click_drag_test
convex_hull_2d
convexhull_export_stl
convexhull_tutorial
coord_tester
cube_test
detectStringChange
grid_test
grid_test2
grid3D_v01
grid3D_v02
grid3d_v03
grid3d_v04
hull_test
hull3d_test
hull3d_test2
laser_art
laser_art_hull
laser_art_objects
libraries
mesh_lib_test
MeshLibDemo
modelabP...A_controlp5
modelabP...ylinderMesh
modelabP..._3D_Terrain
modelabP...icLibs_Mesh
mouse_dra...tation_test
my_example
myCircleTest
mySimpleRead
obj_kinect_01
obj_kinect_01.zip
obj_kinect_02
obj_kinect_2_25.zip

anarV20RW
controlP5
guicomponents
hull3d
mesh
mylib
newhull
newhull.zip
OBJLoader
OBJLoader.zip
openkinect
quickhull3d
quickhull3d.zip
toxiclibs-c...lete-0020
unlekkerLib

library
README

newhull.jar

1 item, 376.06 GB available

Processing: Proclipsing (I)

- You're not stuck in the Processing IDE
- You can program Processing in Eclipse by:

  (1) importing the processing.core library into Eclipse

  Instructions here: http://processing.org/learning/eclipse/

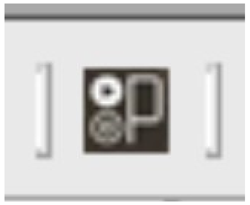
  (2) installing the Proclipsing plugin for Eclipse

  Instructions here:

  http://code.google.com/p/proclipsing/wiki/GettingStarted

Processing: Proclipsing (II)

- When you're done, you should see the Processing icon in the top toolbar →



- Clicking on it will export your Processing sketch into a standalone applet
- To create a new Processing project in Eclipse, go to New → Project → Processing → Processing Project
- Click next - this will allow you to include any core or contributed libraries in your build

Processing: Proclipsing (III)

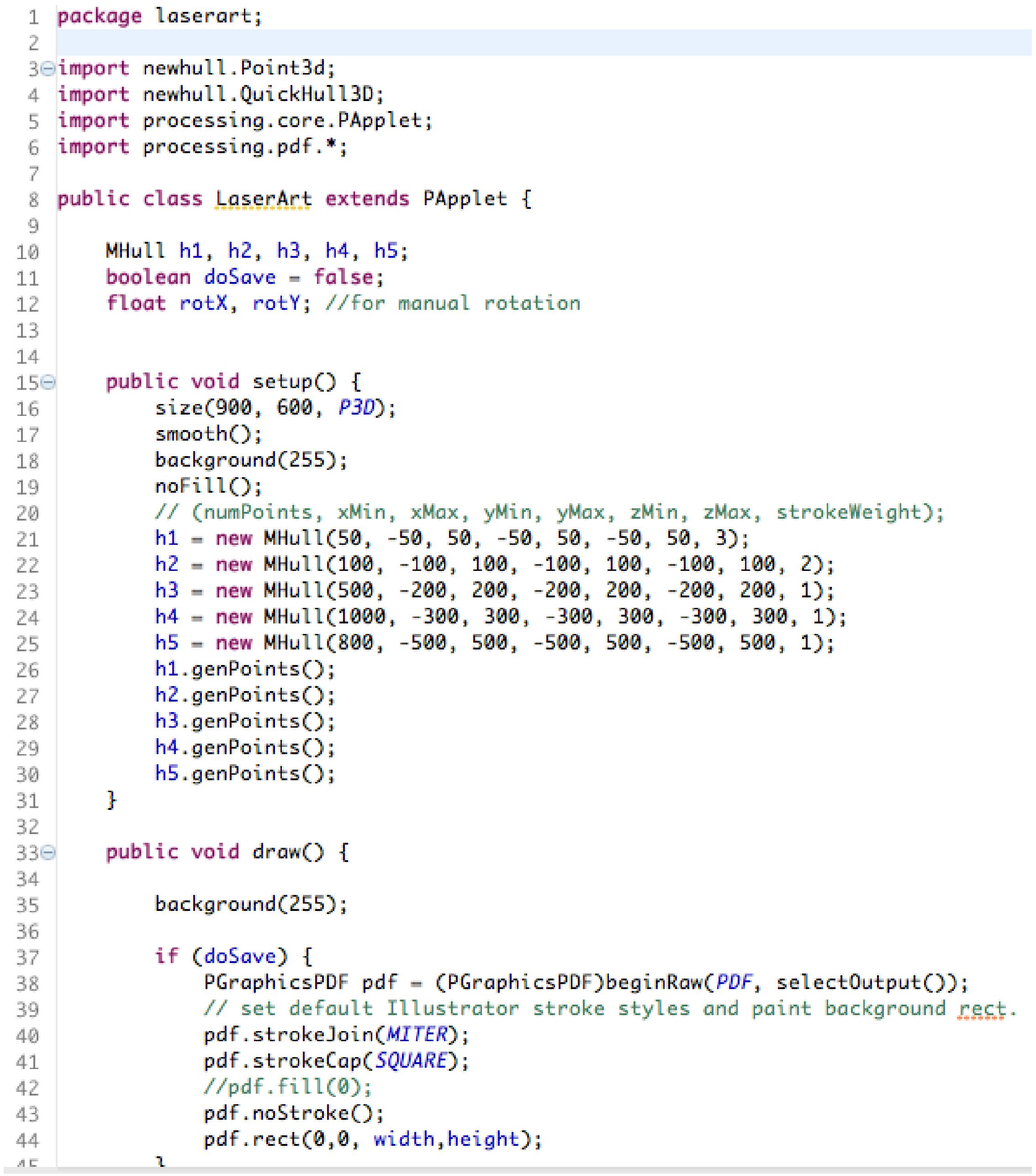
Switching to Eclipse does mean that you have to make a few changes:

Every Processing sketch will now extend PApplet (the core Processing class).

Scope declarations are now required for classes and methods.

To run your sketch, go to Run → Run As → Java Applet

A good overview of getting started with Processing in Eclipse: http://processing.org/learning/eclipse/



```
package laserart;

import newhull.Point3d;
import newhull.QuickHull3D;
import processing.core.PApplet;
import processing.pdf.*;

public class LaserArt extends PApplet {

    MHull h1, h2, h3, h4, h5;
    boolean doSave = false;
    float rotX, rotY; //for manual rotation


    public void setup() {
        size(900, 600, P3D);
        smooth();
        background(255);
        noFill();
        // (numPoints, xMin, xMax, yMin, yMax, zMin, zMax, strokeWeight);
        h1 = new MHull(50, -50, 50, -50, 50, -50, 50, 3);
        h2 = new MHull(100, -100, 100, -100, 100, -100, 100, 2);
        h3 = new MHull(500, -200, 200, -200, 200, -200, 200, 1);
        h4 = new MHull(1000, -300, 300, -300, 300, -300, 300, 1);
        h5 = new MHull(800, -500, 500, -500, 500, -500, 500, 1);
        h1.genPoints();
        h2.genPoints();
        h3.genPoints();
        h4.genPoints();
        h5.genPoints();
    }

    public void draw() {

        background(255);

        if (doSave) {
            PGraphicsPDF pdf = (PGraphicsPDF)beginRaw(PDF, selectOutput());
            // set default Illustrator stroke styles and paint background rect.
            pdf.strokeJoin(MITER);
            pdf.strokeCap(SQUARE);
            //pdf.fill(0);
            pdf.noStroke();
            pdf.rect(0,0, width,height);
        }
```

Processing: Further Resources

- Processing Website – has some examples, tutorials, and an active forum (http://processing.org)
- Open Processing – Tons of example sketches and source code uploaded by users (http://openprocessing.org)
- Learning Processing – A great website by Dan Shiffman with examples and tutorials (http://learningprocessing.com)
- Creative Applications – Another website with some inspirational examples of projects using processing(http://www.creativeapplications.net/category/processing/)

Good luck!

Executive Summary

- Processing is a free, open-source, Java-based framework and IDE that focuses on usability, creativity, and extensibility
- Processing has a large base of included and contributed libraries that extend its functionality into video, audio, hardware communication, file i/o, math and physics simulation, advanced 3D, and more
- Processing supports all Java-like object oriented definitions and techniques, as well as allowing use of standard and custom Java libraries within processing itself
- Processing can also be used in Eclipse using the Proclipsing plugin or by importing the Core Processing libraries into Eclipse