

Programming in Java

with Processing

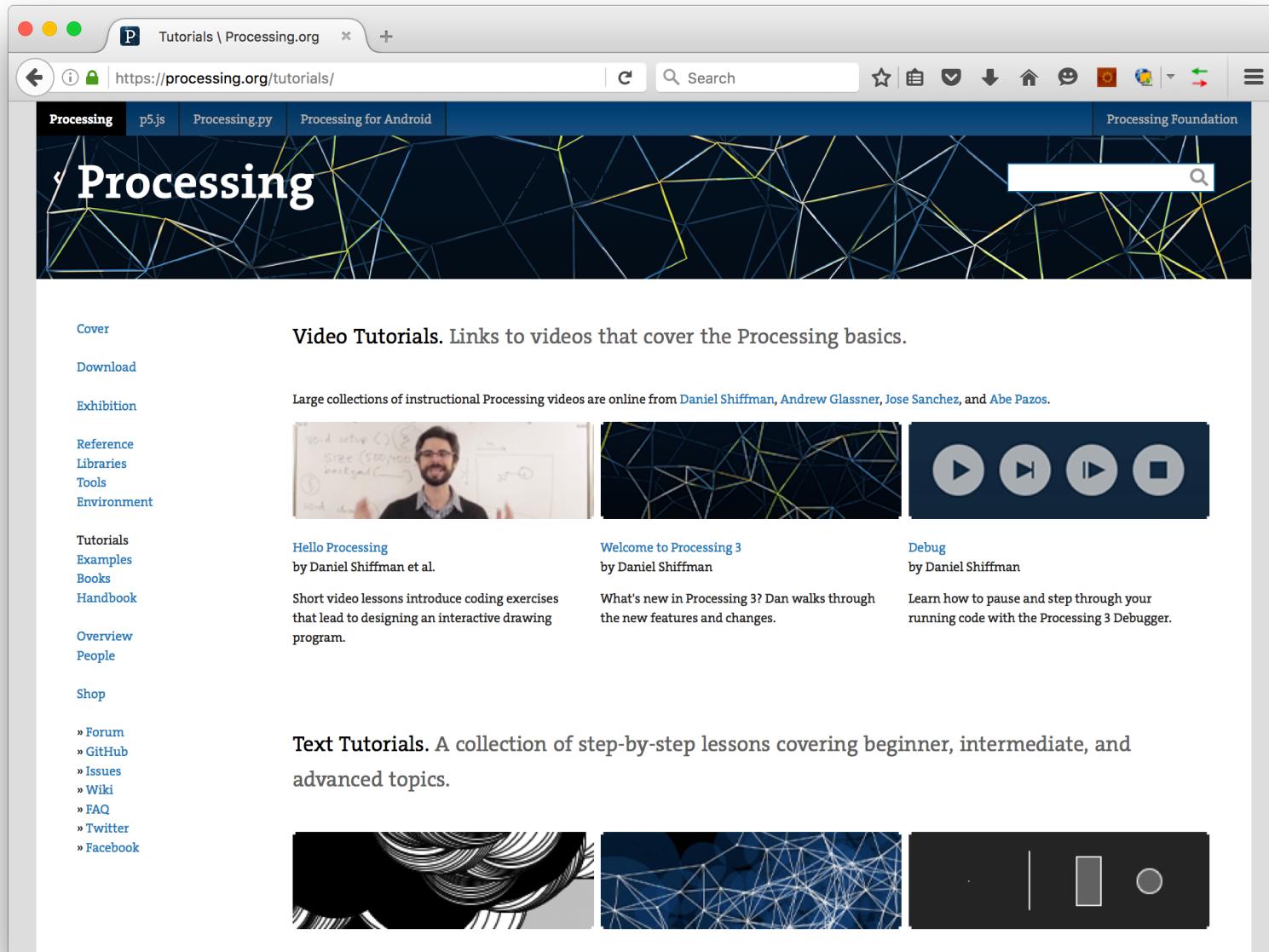
https://processing.org/

The screenshot shows a web browser displaying the official Processing website at <https://processing.org>. The page features a dark background with a complex geometric wireframe pattern. At the top, there's a navigation bar with tabs for "Processing", "p5.js", "Processing.py", "Processing for Android", and "Processing Foundation". A search bar is located in the top right corner.

The main content area includes:

- A sidebar on the left with links to "Cover", "Download", "Exhibition", "Reference", "Libraries", "Tools", "Environment", "Tutorials", "Examples", "Books", "Handbook", "Overview", "People", "Shop", and social media links for "Forum", "GitHub", "Issues", "Wiki", "FAQ", "Twitter", and "Facebook".
- A central video player showing a thumbnail for "Welcome to Processing 3" by Dan Shiffman. The thumbnail features a man interacting with a large screen displaying a colorful, abstract visualization. Below the video is a caption: "Welcome to Processing 3! Dan explains the new features and changes; the links Dan mentions are on the [Vimeo page](#)".
- Three exhibition cards on the right:
 - "cf.city flows" by Till Nagel and Christopher Pietsch, featuring a thumbnail of a complex network visualization.
 - "Automatic Orchestra" by Students and Lecturers at Digital Media Bremen, featuring a thumbnail of a green grid-based visualization.
 - "Possible, Plausible, Potential" by Miguel Nóbrega, featuring a thumbnail of a colorful, geometric 3D scene.
- Links at the bottom of the sidebar: "» Free to download and open source".

https://processing.org/tutorials/



The screenshot shows the 'Tutorials' section of the Processing.org website. At the top, there's a navigation bar with tabs for 'Processing', 'p5.js', 'Processing.py', 'Processing for Android', and 'Processing Foundation'. Below the navigation is a large banner featuring the word 'Processing' in white on a dark background with a complex geometric pattern of blue and yellow lines. A search bar is located in the top right corner of the banner.

Cover

Video Tutorials. Links to videos that cover the Processing basics.

Large collections of instructional Processing videos are online from [Daniel Shiffman](#), [Andrew Glassner](#), [Jose Sanchez](#), and [Abe Pazos](#).

Download

Exhibition

Reference

Libraries

Tools

Environment

Tutorials

Examples

Books

Handbook

Overview

People

Shop

[» Forum](#)

[» GitHub](#)

[» Issues](#)

[» Wiki](#)

[» FAQ](#)

[» Twitter](#)

[» Facebook](#)

Hello Processing
by Daniel Shiffman et al.

Welcome to Processing 3
by Daniel Shiffman

Debug
by Daniel Shiffman

Short video lessons introduce coding exercises that lead to designing an interactive drawing program.

What's new in Processing 3? Dan walks through the new features and changes.

Learn how to pause and step through your running code with the Processing 3 Debugger.

Text Tutorials. A collection of step-by-step lessons covering beginner, intermediate, and advanced topics.

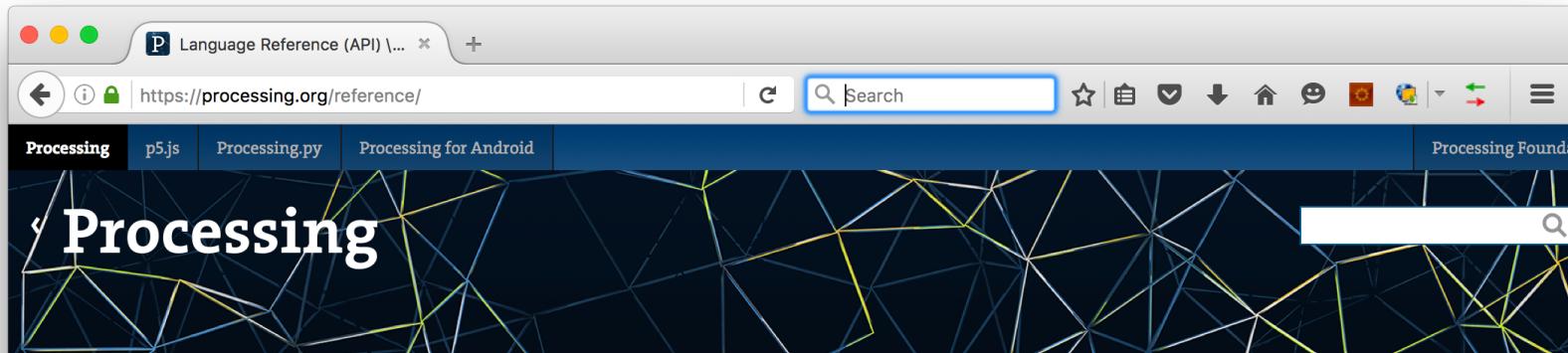
  

Getting Started

Processing Overview

Coordinates, Colors and Classes

https://processing.org/reference/



The screenshot shows a web browser window displaying the Processing Language Reference (API) at <https://processing.org/reference/>. The page features a dark background with a geometric wireframe pattern. The title 'Processing' is prominently displayed on the left. A search bar is at the top right, along with various browser navigation and utility icons.

Cover **Reference.** Processing was designed to be a flexible software sketchbook.

Download

Exhibition

Structure

- Reference: () (parentheses)
- Libraries: , (comma)
- Tools: . (dot)
- Environment: /* */ (multiline comment)
- Tutorials: /** */ (doc comment)
- Examples: // (comment)
- Books: ; (semicolon)
- Handbook: = (assign)
- Overview: [] (array access)
- People: {} (curly braces)
- Shop: catch
- » Forum: class
- » GitHub: draw()
- » Issues: exit()
- » Wiki: extends
- » FAQ: false
- » Twitter: final
- » Facebook: implements
- Import: import
- Loop: loop()
- New: new
- NoLoop: noLoop()
- Null: null

Shape

- createShape()
- loadShape()
- PShape
- 2D Primitives:
 - arc()
 - ellipse()
 - line()
 - point()
 - quad()
 - rect()
 - triangle()
- Curves:
 - bezier()
 - bezierDetail()
 - bezierPoint()
 - bezierTangent()
 - curve()
 - curveDetail()
 - curvePoint()
 - curveTangent()
 - curveTightness()

Color

- Setting
- background()
- clear()
- colorMode()
- fill()
- noFill()
- noStroke()
- stroke()
- Creating & Reading
 - alpha()
 - blue()
 - brightness()
 - color()
 - green()
 - hue()
 - lerpColor()
 - red()
 - saturation()
- Image
 - createImage()

https://processing.org/reference/arc_.html [popStyle\(\)](#)

Name**println()****Examples**

```
String s = "The size is ";
int w = 1920;
int h = 1080;
println(s);
println(w, "x", h);

// This program writes to the console:
// The size is
// 1920 x 1080
```

```
print("begin- ");
float f = 0.3;
int i = 1024;
print("f is " + f + " and i is " + 1024);
String s = " -end";
println(s);

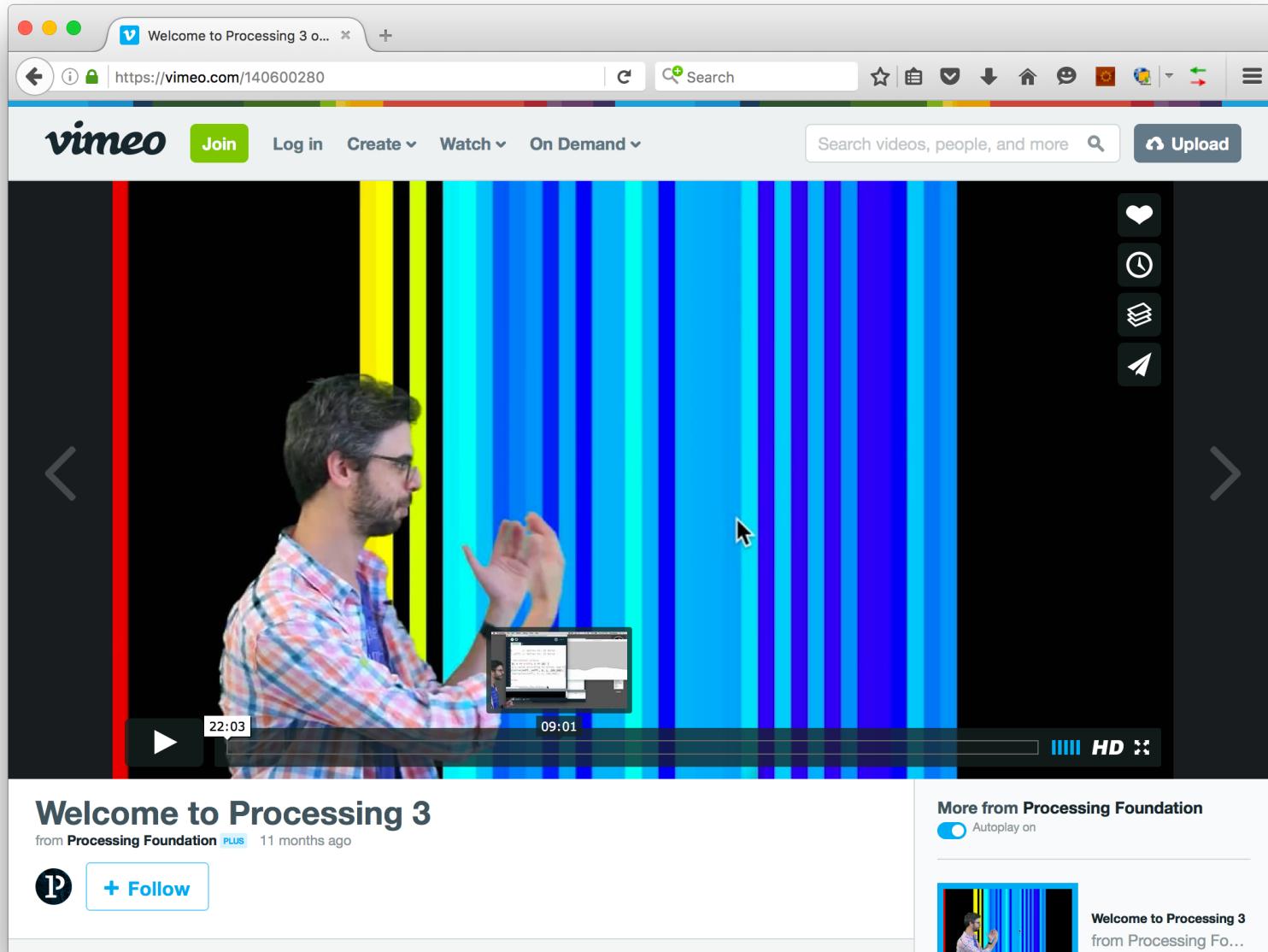
// This program writes to the console:
// "begin- f is 0.3 and i is 1024 -end"
```

Description

The `println()` function writes to the console area, the black rectangle at the bottom of the Processing environment. This function is often helpful for looking at the data a program is producing. Each call to this function creates a new line of output. More than one parameter can be passed into the function by separating them with commas. Alternatively, individual elements can be separated with quotes ("") and joined with the addition operator (+).

Before Processing 2.1, `println()` was used to write array data to the console. Now, use `printArray()` to write array data to the console.

<https://vimeo.com/140600280>



P Tutorials | Processing.org

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

» Forum
» GitHub
» Issues
» Wiki
» FAQ
» Twitter
» Facebook

Text Tutorials. A collection of step-by-step lessons covering beginner, intermediate, and advanced topics.



Getting Started
by Casey Reas and Ben Fry

Welcome to Processing! This introduction covers the basics of writing Processing code.

Level: Beginner



Color
by Daniel Shiffman

An introduction to digital color.

Level: Beginner



Objects
by Daniel Shiffman

The basics of object-oriented programming.

Level: Beginner



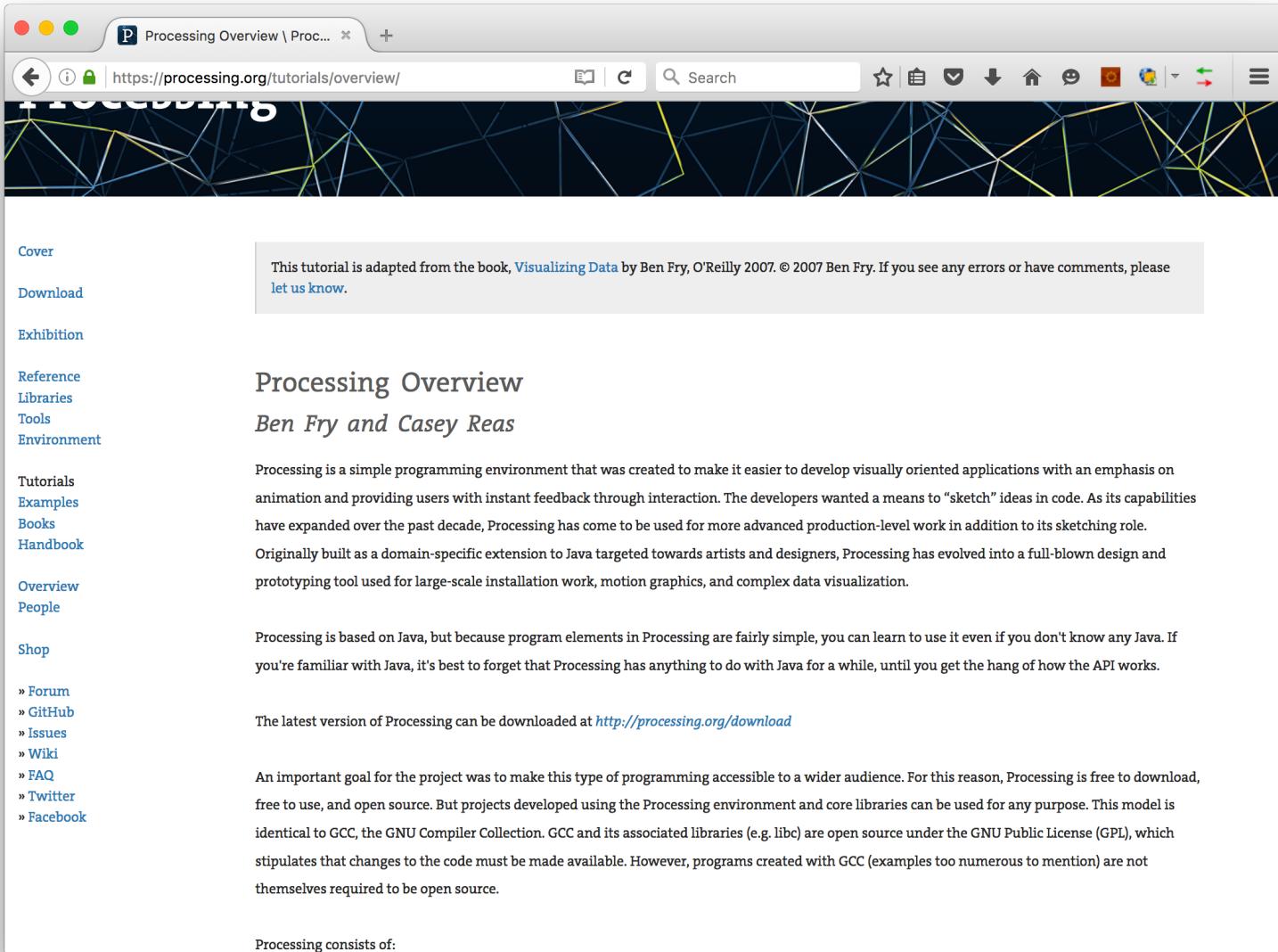
Interactivity
by Casey Reas and Ben Fry

Introduction to interactivity with the mouse and keyboard.

Level: Beginner



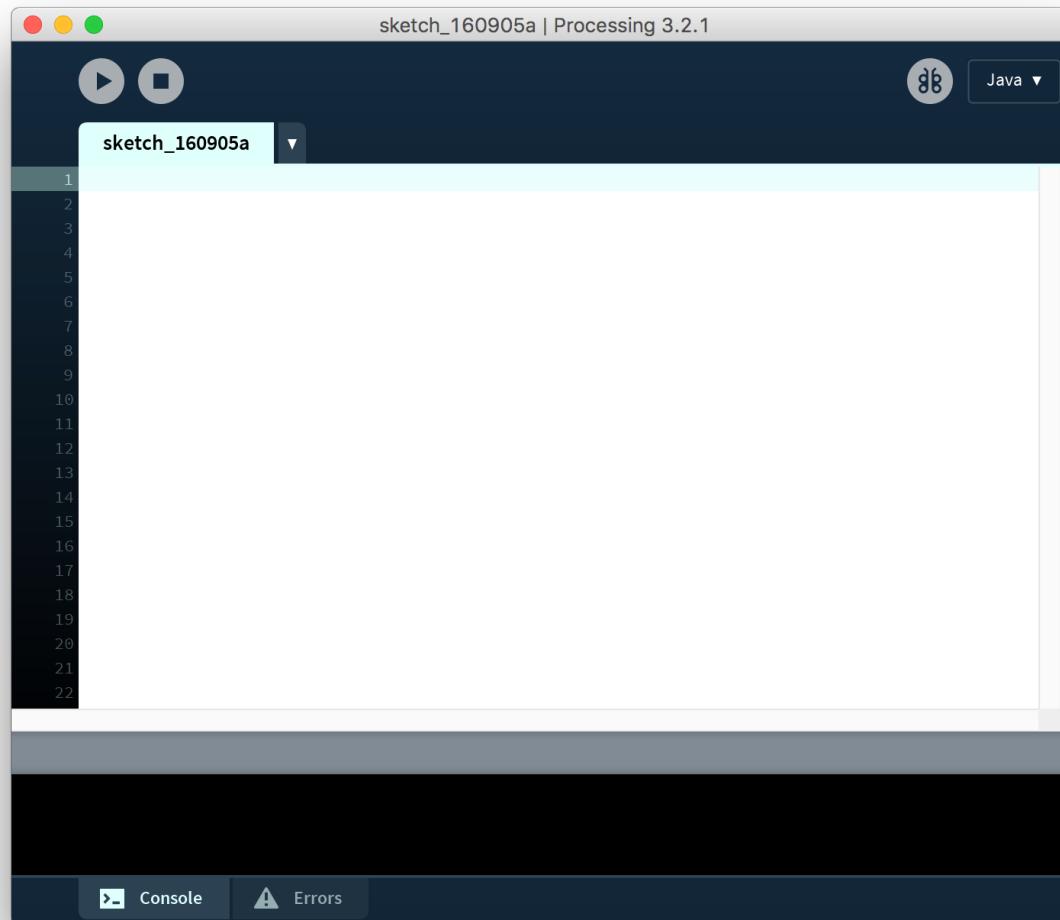
Processing Overview



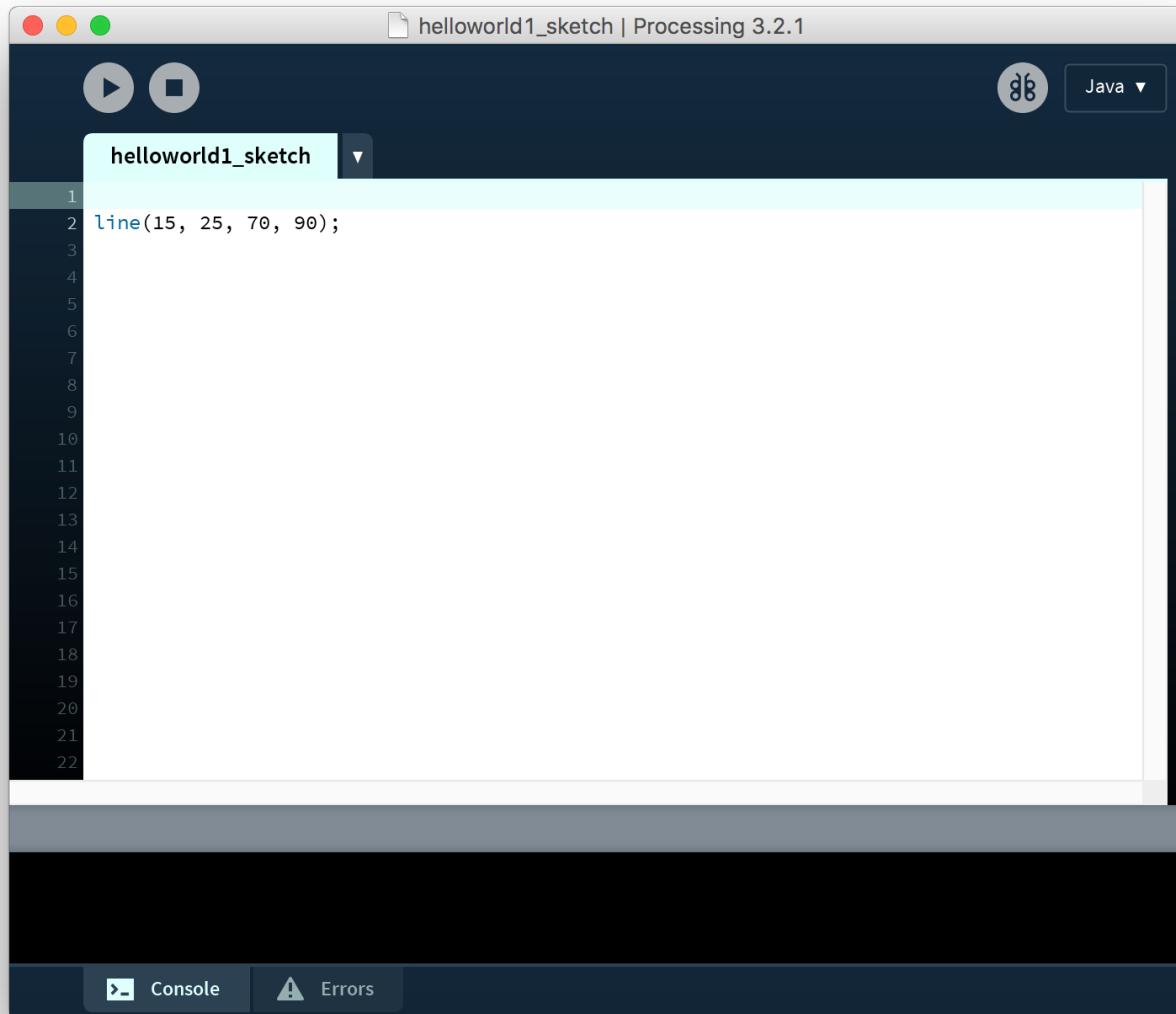
The screenshot shows a web browser window displaying the 'Processing Overview' page from <https://processing.org/tutorials/overview/>. The page features a dark background with a complex geometric pattern of blue and yellow lines. On the left, there's a sidebar with links like 'Cover', 'Download', 'Exhibition', 'Reference', 'Libraries', 'Tools', 'Environment', 'Tutorials', 'Examples', 'Books', 'Handbook', 'Overview', 'People', 'Shop', and social media links for 'Forum', 'GitHub', 'Issues', 'Wiki', 'FAQ', 'Twitter', and 'Facebook'. The main content area has a heading 'Processing Overview' and a subtitle 'Ben Fry and Casey Reas'. It includes a note about the tutorial being adapted from the book 'Visualizing Data' by Ben Fry. Below this, there's a detailed paragraph about Processing's history and evolution, followed by another paragraph about its Java-based nature and accessibility. At the bottom, there's a section titled 'Processing consists of:'.

[**https://processing.org/tutorials/overview/**](https://processing.org/tutorials/overview/)

Processing 3 | Interactive IDE



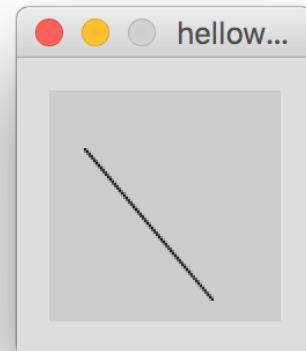
Simple Hello World



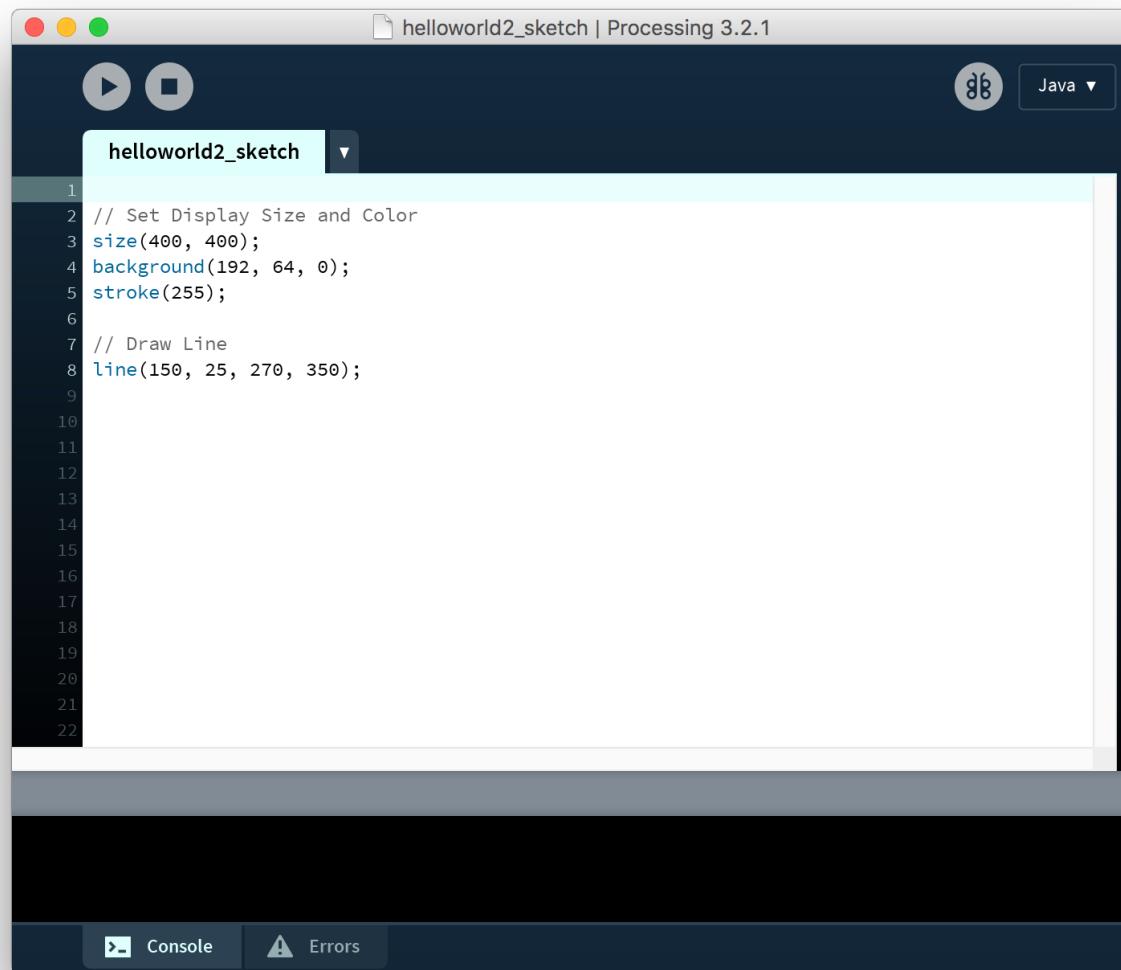
The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld1_sketch | Processing 3.2.1". The main window displays the code for "helloworld1_sketch" with the following content:

```
1 line(15, 25, 70, 90);
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

The code consists of two lines: line 1 contains the function call "line(15, 25, 70, 90);". The Processing environment includes standard controls like play/pause buttons and a Java dropdown menu. At the bottom, there are tabs for "Console" and "Errors".



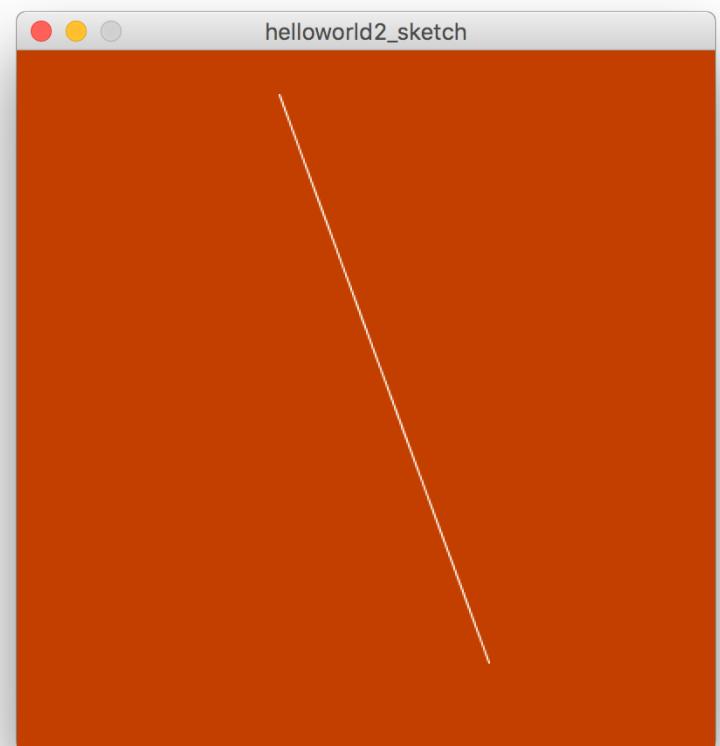
Another Hello World



The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld2_sketch | Processing 3.2.1". The main window displays the following Java code:

```
1 // Set Display Size and Color
2 size(400, 400);
3 background(192, 64, 0);
4 stroke(255);
5
6 // Draw Line
7 line(150, 25, 270, 350);
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

The code sets the display size to 400x400 pixels, changes the background color to orange-red (#19A85D), sets the stroke color to black, and draws a single white line from (150, 25) to (270, 350). The IDE has tabs for "Console" and "Errors" at the bottom.

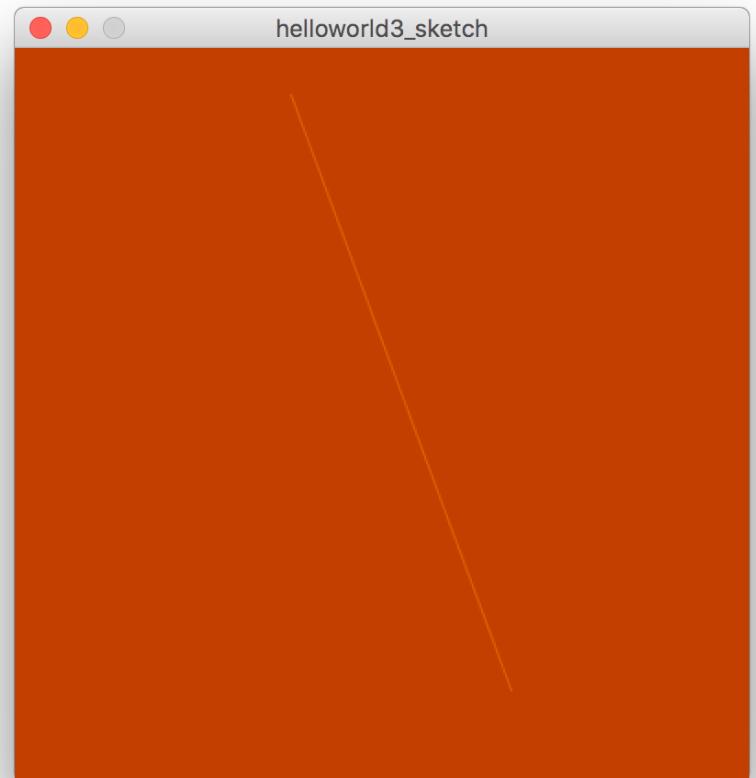


Add Some Style!

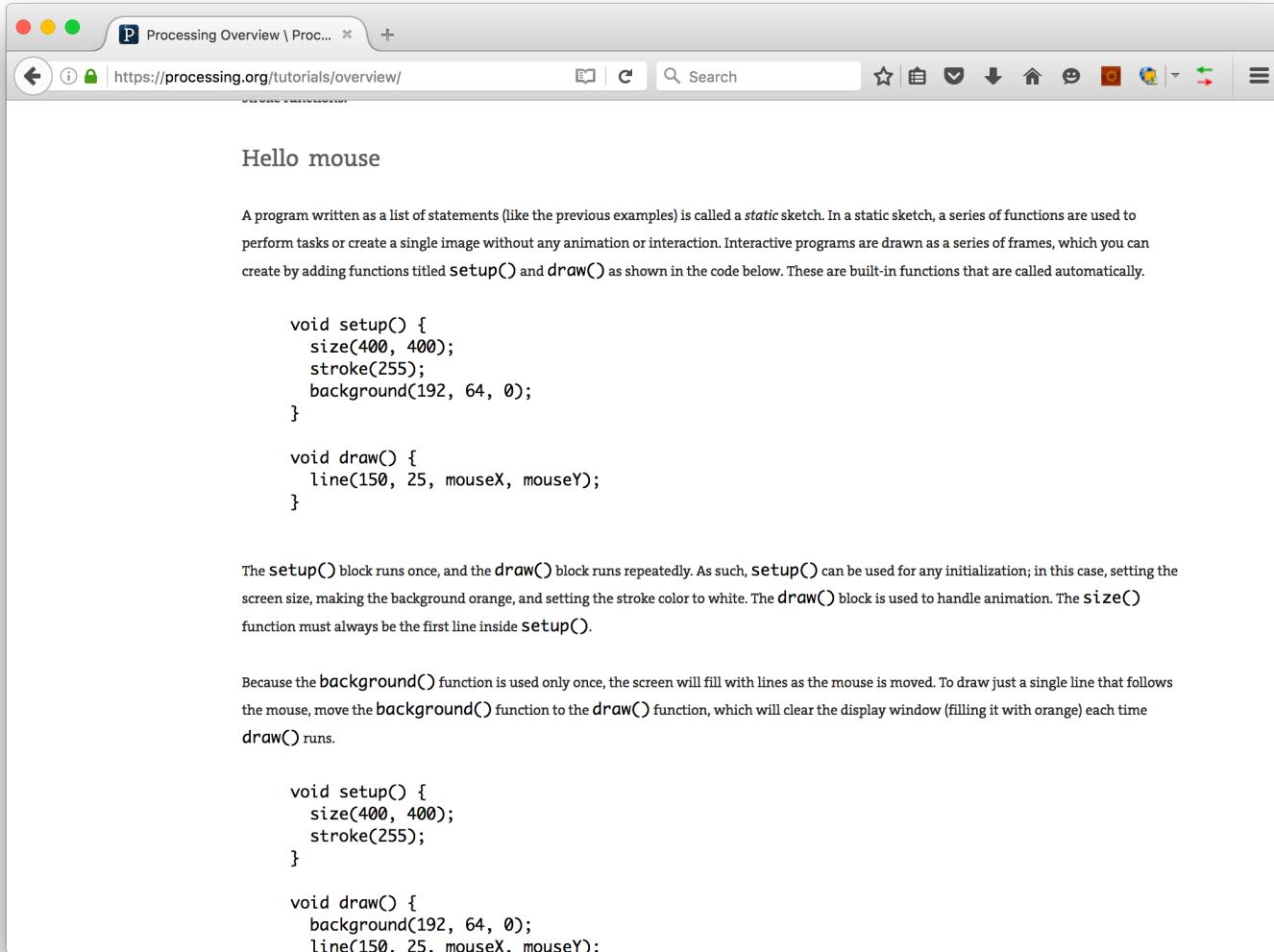
The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld3_sketch | Processing 3.2.1". The code editor contains the following Java code:

```
1 // Set Display Size and Color
2 size(400, 400);
3 background(192, 64, 0);
4 stroke(255);
5
6 // Set Stroke Style
7 stroke(255);           // sets the stroke color to white
8 stroke(255, 255, 255); // identical to the line above
9 stroke(255, 128, 0);   // bright orange (red 255, green 128, blue 0)
10 stroke(#FF8000);       // bright orange as a web color
11 stroke(255, 128, 0, 128); // bright orange with 50% transparency
12
13 // Draw Line
14 line(150, 25, 270, 350);
15
16
17
18
19
20
21
22
```

The status bar at the bottom left says "Done saving." The bottom navigation bar has tabs for "Console" and "Errors".



Hello Mouse

A screenshot of a web browser window displaying the "Processing Overview" tutorial page from processing.org. The title "Hello mouse" is visible at the top. Below it, a paragraph explains the difference between static sketches and interactive programs, mentioning the setup() and draw() functions. A code block shows the Pseudocode for the Hello Mouse sketch. Another paragraph describes how the setup() function runs once and draw() runs repeatedly, with a note about moving the background() call to draw(). A final code block shows the complete Pseudocode for the sketch.

Hello mouse

A program written as a list of statements (like the previous examples) is called a *static* sketch. In a static sketch, a series of functions are used to perform tasks or create a single image without any animation or interaction. Interactive programs are drawn as a series of frames, which you can create by adding functions titled `setup()` and `draw()` as shown in the code below. These are built-in functions that are called automatically.

```
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}
```

The `setup()` block runs once, and the `draw()` block runs repeatedly. As such, `setup()` can be used for any initialization; in this case, setting the screen size, making the background orange, and setting the stroke color to white. The `draw()` block is used to handle animation. The `size()` function must always be the first line inside `setup()`.

Because the `background()` function is used only once, the screen will fill with lines as the mouse is moved. To draw just a single line that follows the mouse, move the `background()` function to the `draw()` function, which will clear the display window (filling it with orange) each time `draw()` runs.

```
void setup() {  
    size(400, 400);  
    stroke(255);  
}  
  
void draw() {  
    background(192, 64, 0);  
    line(150, 25, mouseX, mouseY);  
}
```

<https://processing.org/tutorials/overview/>

Setup and Draw

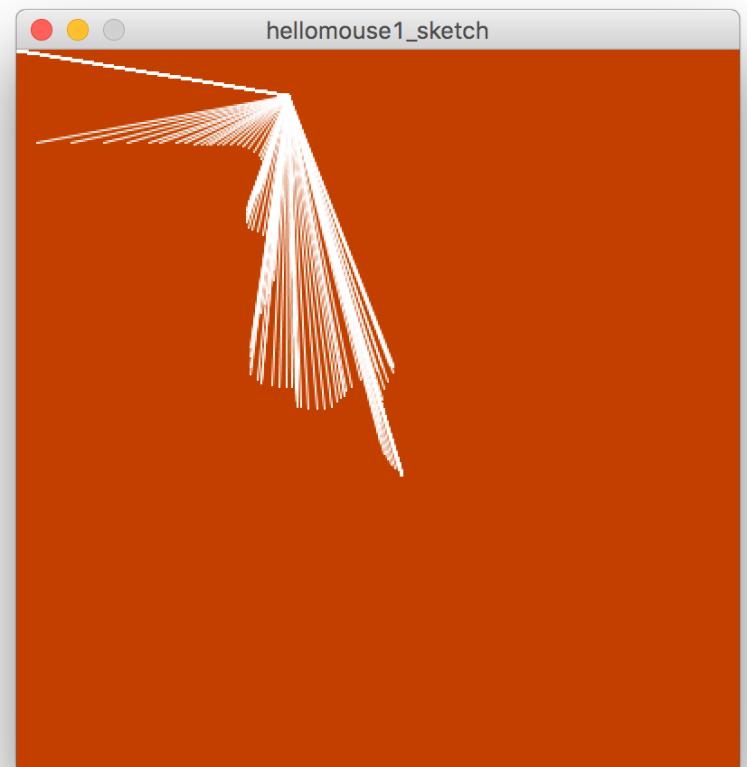
hellomouse1_sketch | Processing 3.2.1

hellomouse1_sketch

```
1 // called only once at start-up
2 void setup() {
3     size(400, 400);
4     stroke(255);
5     background(192, 64, 0);
6 }
7
8 // called repeatedly by processing
9 void draw() {
10    line(150, 25, mouseX, mouseY);
11 }
12 }
```

Done saving.

Console Errors

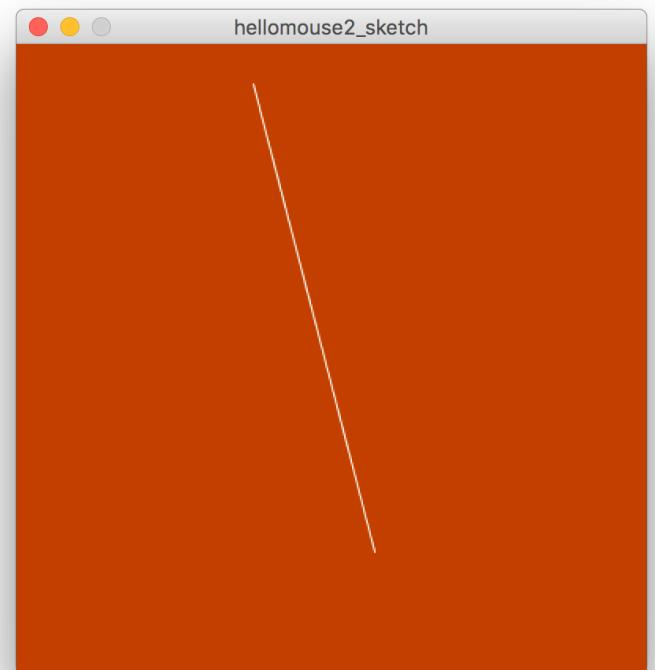


Following Line

The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "hellomouse2_sketch | Processing 3.2.1". Below the title bar are standard window controls (red, yellow, green) and a toolbar with a play button, square button, and a circular icon with two vertical bars. To the right of the toolbar is a "Java" dropdown menu. The main area contains the sketch code:

```
1 // called only once at start-up
2 void setup() {
3     size(400, 400);
4     stroke(255);
5 }
6
7 // called repeatedly by processing
8 // refresh background with each call
9 void draw() {
10    background(192, 64, 0);
11    line(150, 25, mouseX, mouseY);
12 }
13
14
15
16
17
18
19
20
21
22
```

At the bottom of the code editor, a message "Done saving." is displayed. At the very bottom of the IDE, there are tabs for "Console" and "Errors".



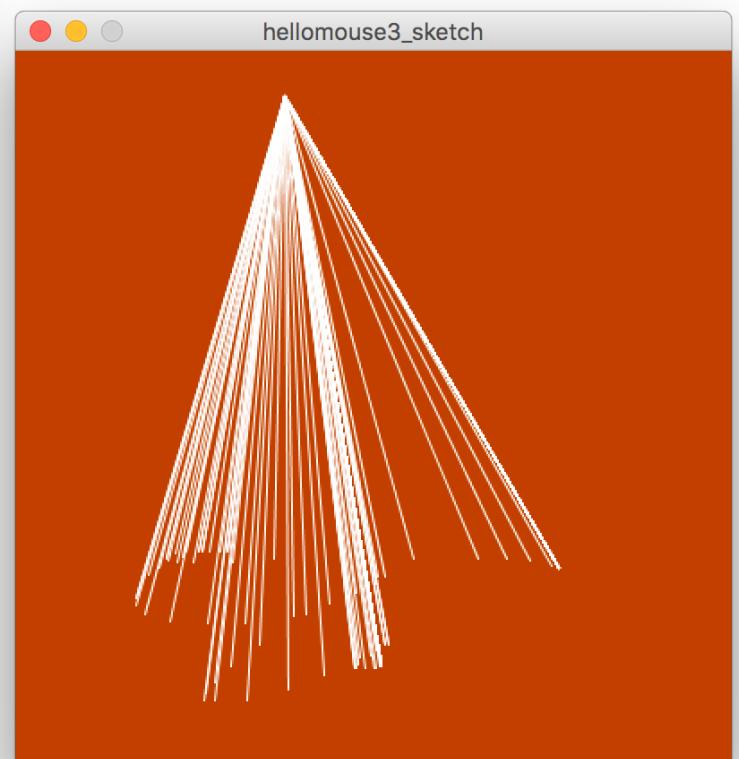
Using Mouse Press

helloworld3_sketch | Processing 3.2.1

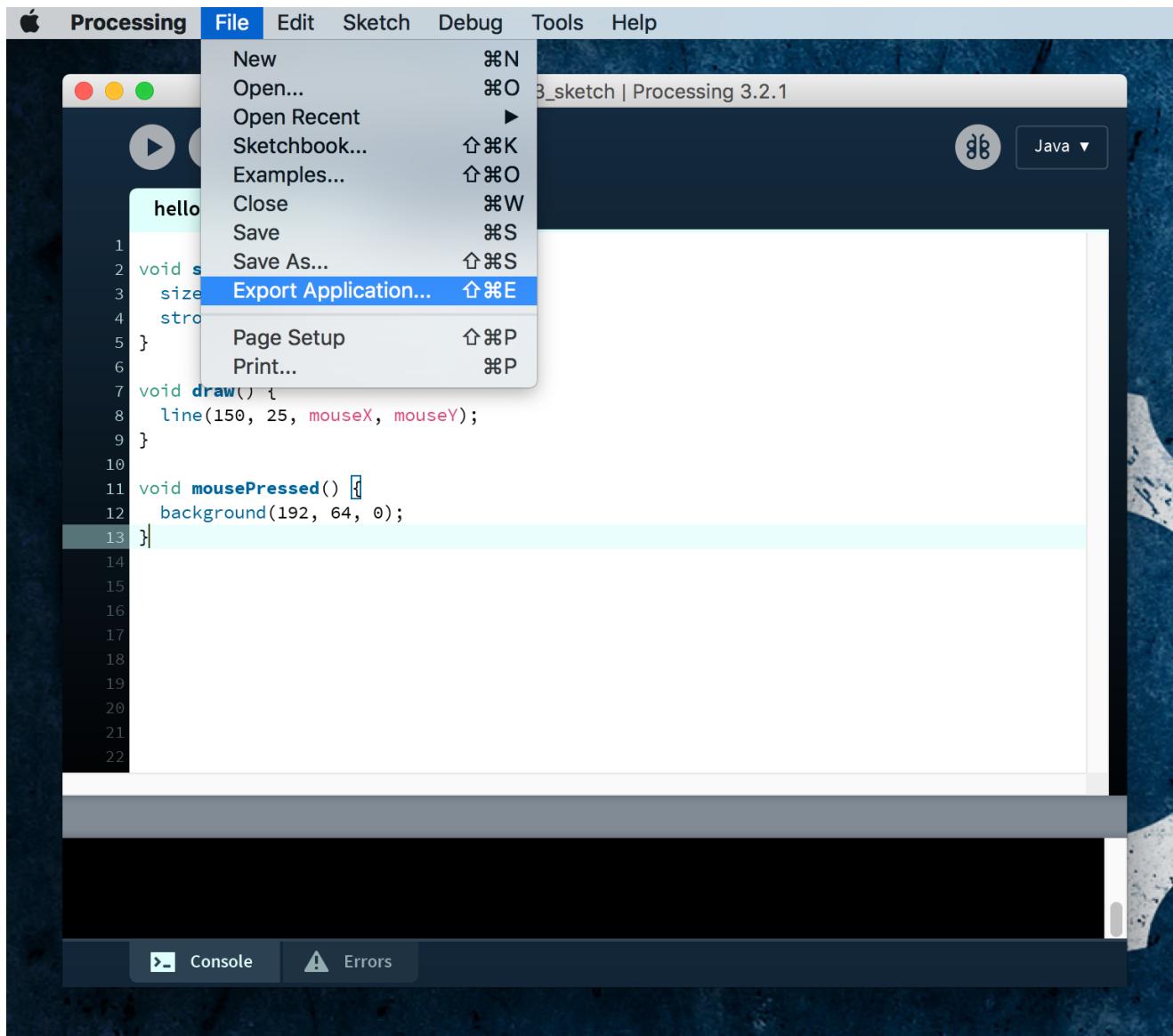
```
1 void setup() {
2     size(400, 400);
3     stroke(255);
4 }
5
6 void draw() {
7     line(150, 25, mouseX, mouseY);
8 }
9
10 void mousePressed() {
11     background(192, 64, 0);
12 }
13
14
15
16
17
18
19
20
21
22
```

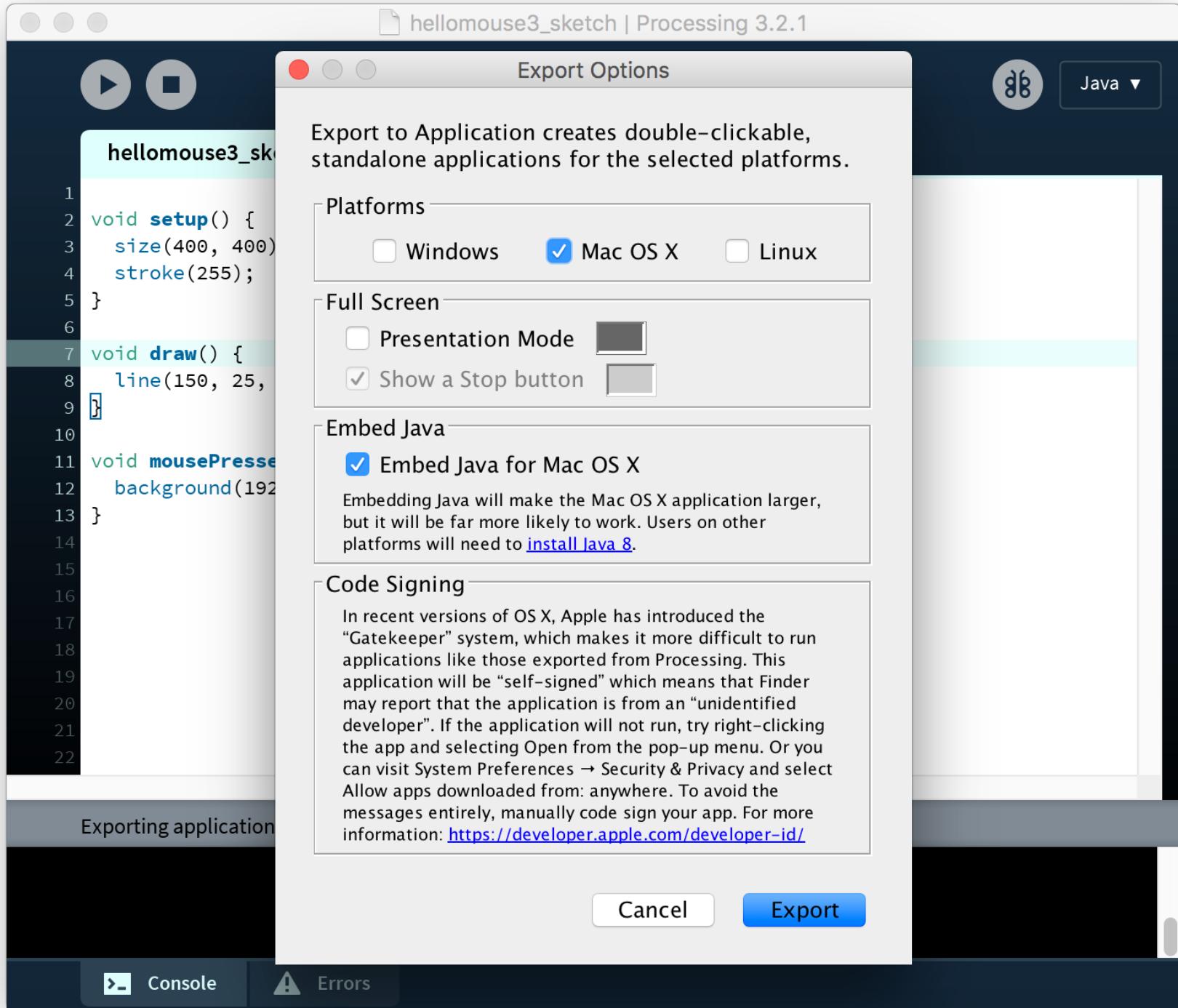
Done saving.

Console Errors

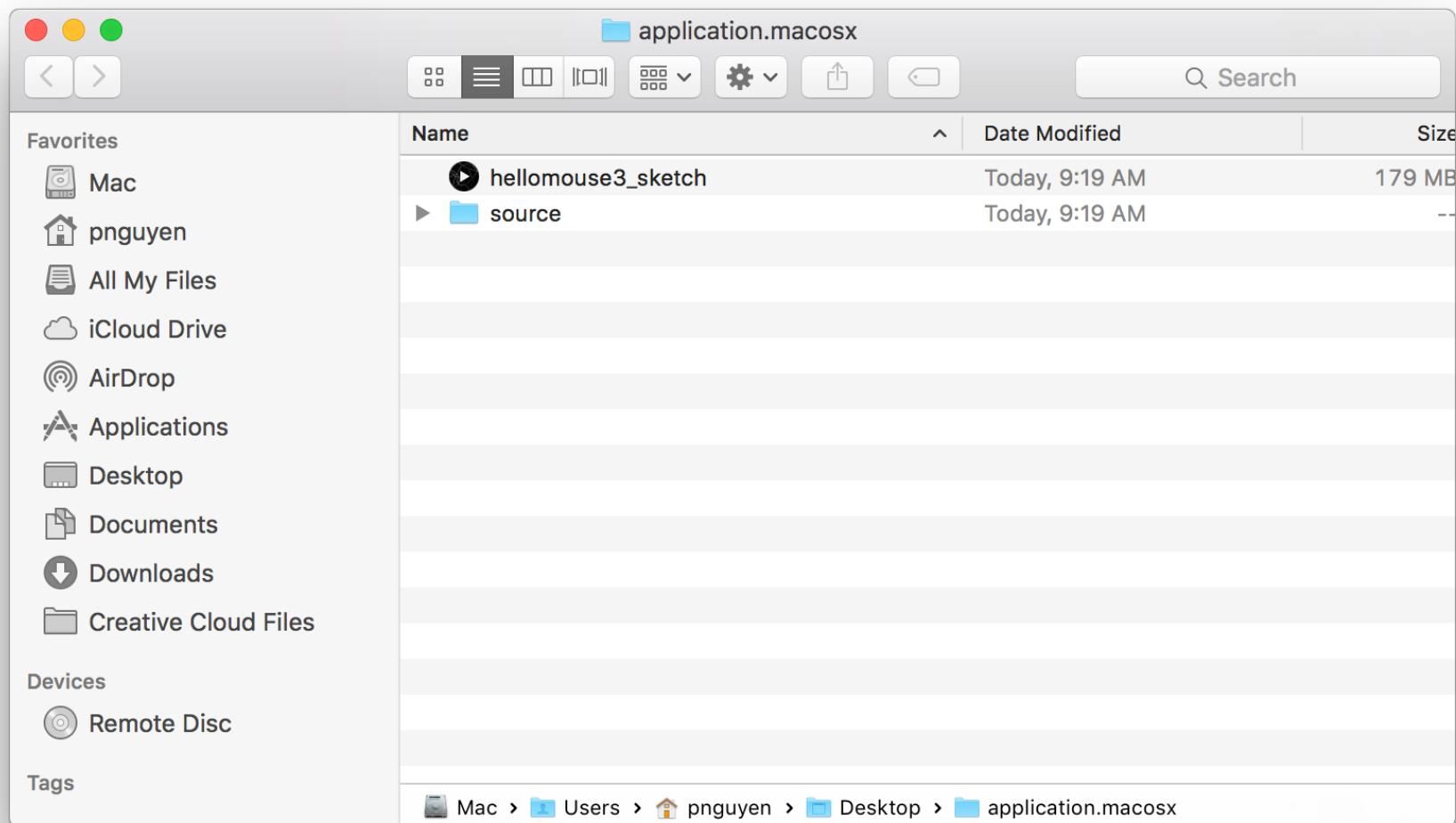


Exporting Application





Exported Application | Click to Run

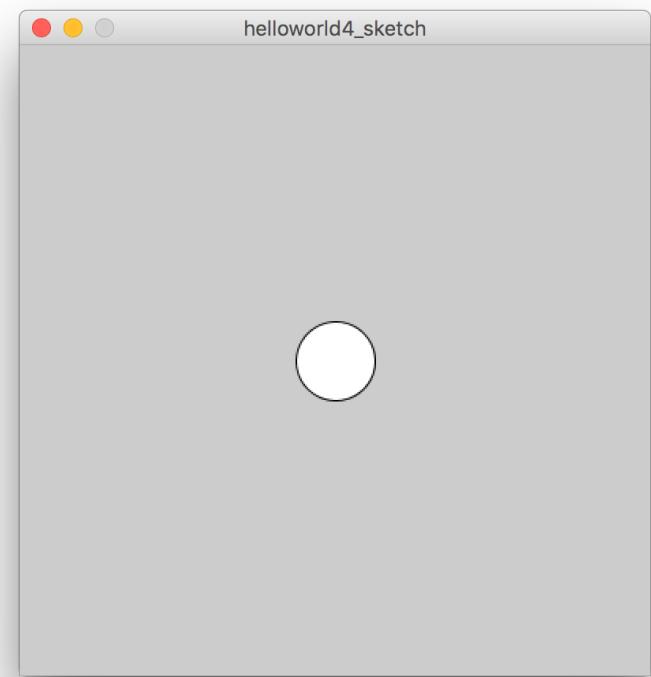


Sizing

The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld4_sketch | Processing 3.2.1". The code editor contains the following Java code:

```
1 size(400, 400);
2
3 // The wrong way to specify the middle of the screen
4 // ellipse(200, 200, 50, 50);
5
6 // Always the middle, no matter how the size() line changes
7 ellipse(width/2, height/2, 50, 50);
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

The code editor has a dark theme with syntax highlighting. The line "ellipse(width/2, height/2, 50, 50);" is highlighted in light blue. At the bottom of the IDE, there are tabs for "Console" and "Errors", with "Console" being the active tab.



helloworld4_sketch | Processing 3.2.1

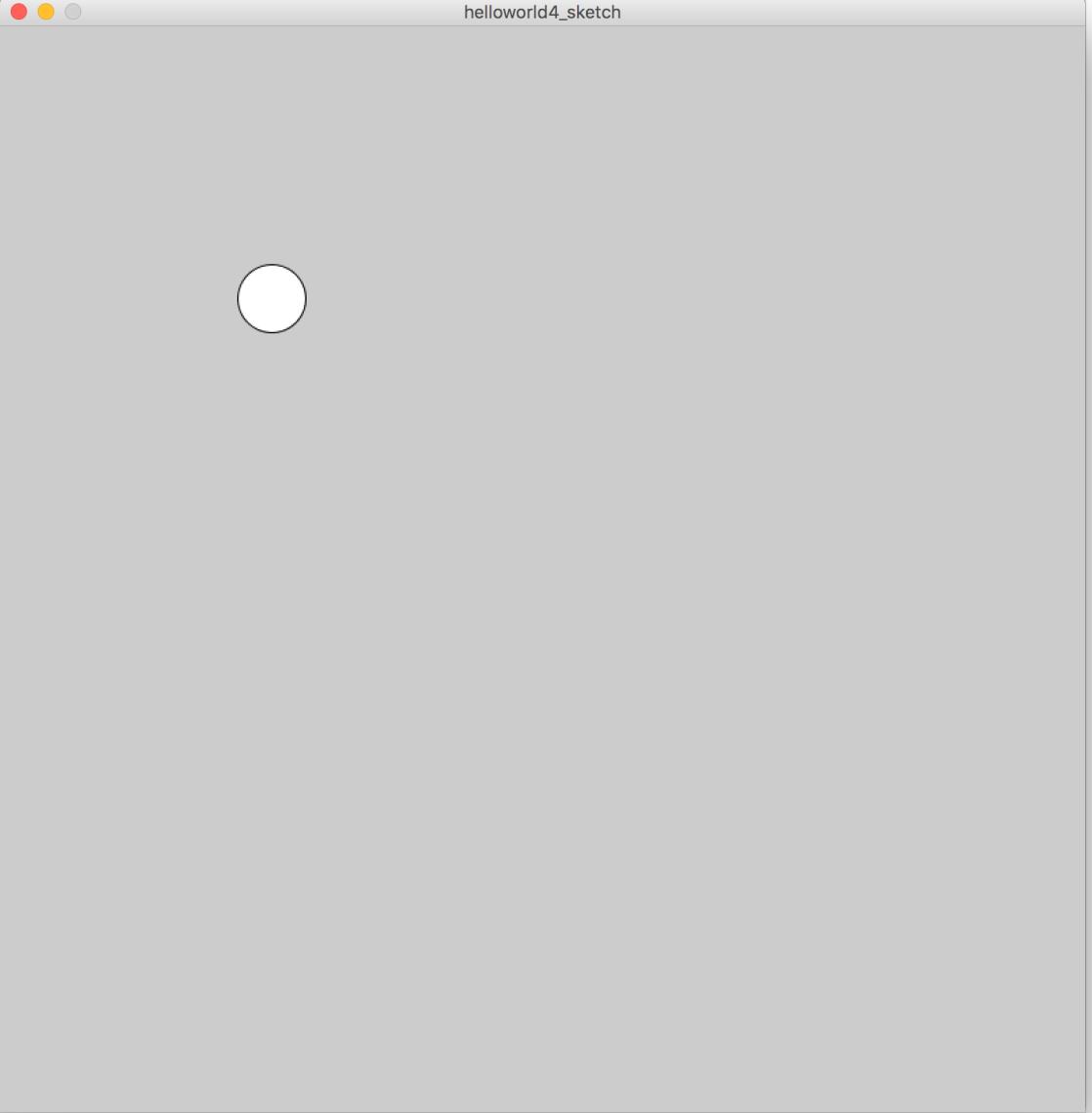
Java ▾

helloworld4_sketch

```
1 size(800, 800);
2 // The wrong way to specify the middle of the screen
3 ellipse(200, 200, 50, 50);
4 // Always the middle, no matter how the size() line changes
5 //ellipse(width/2, height/2, 50, 50);
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

Console Errors

helloworld4_sketch



The image shows the Processing 3.2.1 IDE interface. On the left, the code editor displays a sketch named 'helloworld4_sketch' with the following code:1 size(800, 800);
2 // The wrong way to specify the middle of the screen
3 ellipse(200, 200, 50, 50);
4 // Always the middle, no matter how the size() line changes
5 //ellipse(width/2, height/2, 50, 50);
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

On the right, the preview window titled 'helloworld4_sketch' shows a single white circle centered at the top of a gray background. The preview window has a title bar with red, yellow, and green buttons, and a close button in the top-left corner.

Loading Data & Images

Loading and displaying data

One of the unique aspects of the Processing API is the way files are handled. The `loadImage()` and `loadStrings()` functions each expect to find a file inside a folder named *data*, which is a subdirectory of the sketch folder.

Advanced Topic: Notes on the data folder

The *data* folder addresses a common frustration when dealing with code that is tested locally but deployed over the web. Like Java, software written with Processing is subject to security restrictions that determine how a program can access resources such as the local hard disk or other servers via the Internet. This prevents malicious developers from writing code that could harm your computer or compromise your data.

The security restrictions can be tricky to work with during development. When running a program locally, data can be read directly from the disk, though it must be placed relative to the user's "working directory," generally the location of the application. When running online, data must come from a location on the same server. It might be bundled with the code itself (in a JAR archive, discussed later; or from another URL on the same server). For a local file, Java's `FileInputStream` class can be used. If the file is bundled in a JAR archive, the `getResource()` function is used. For a file on the server, `URL.openStream()` might be employed. During the journey from development to deployment, it may be necessary to use all three of these functions.

With Processing, each of these scenarios (and some others) is handled transparently by the file API functions. By placing resources in the *data* folder, Processing packages the files as necessary for online and offline use.

helloworld5_sketch | Processing 3.2.1

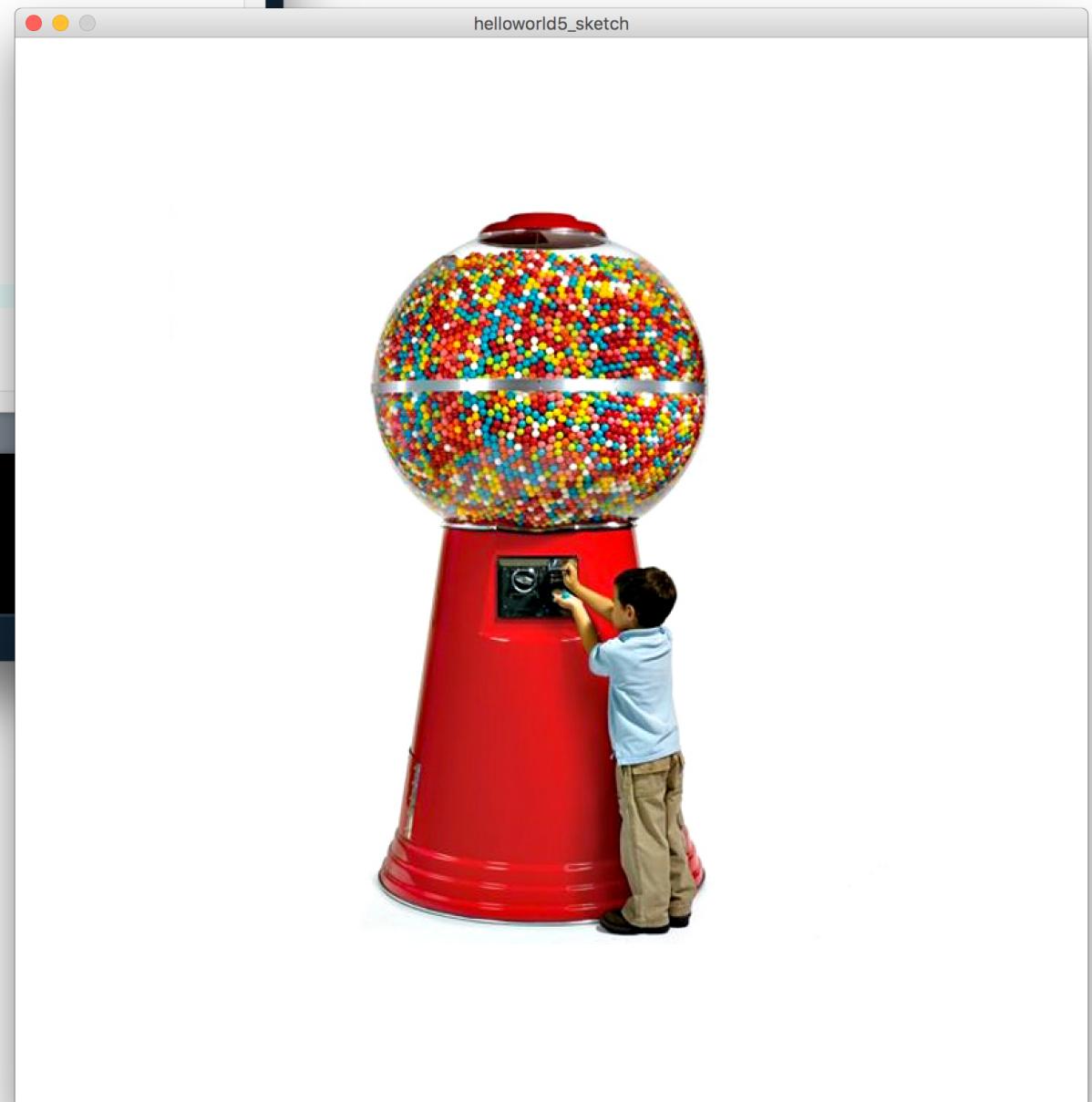
Java ▾

helloworld5_sketch

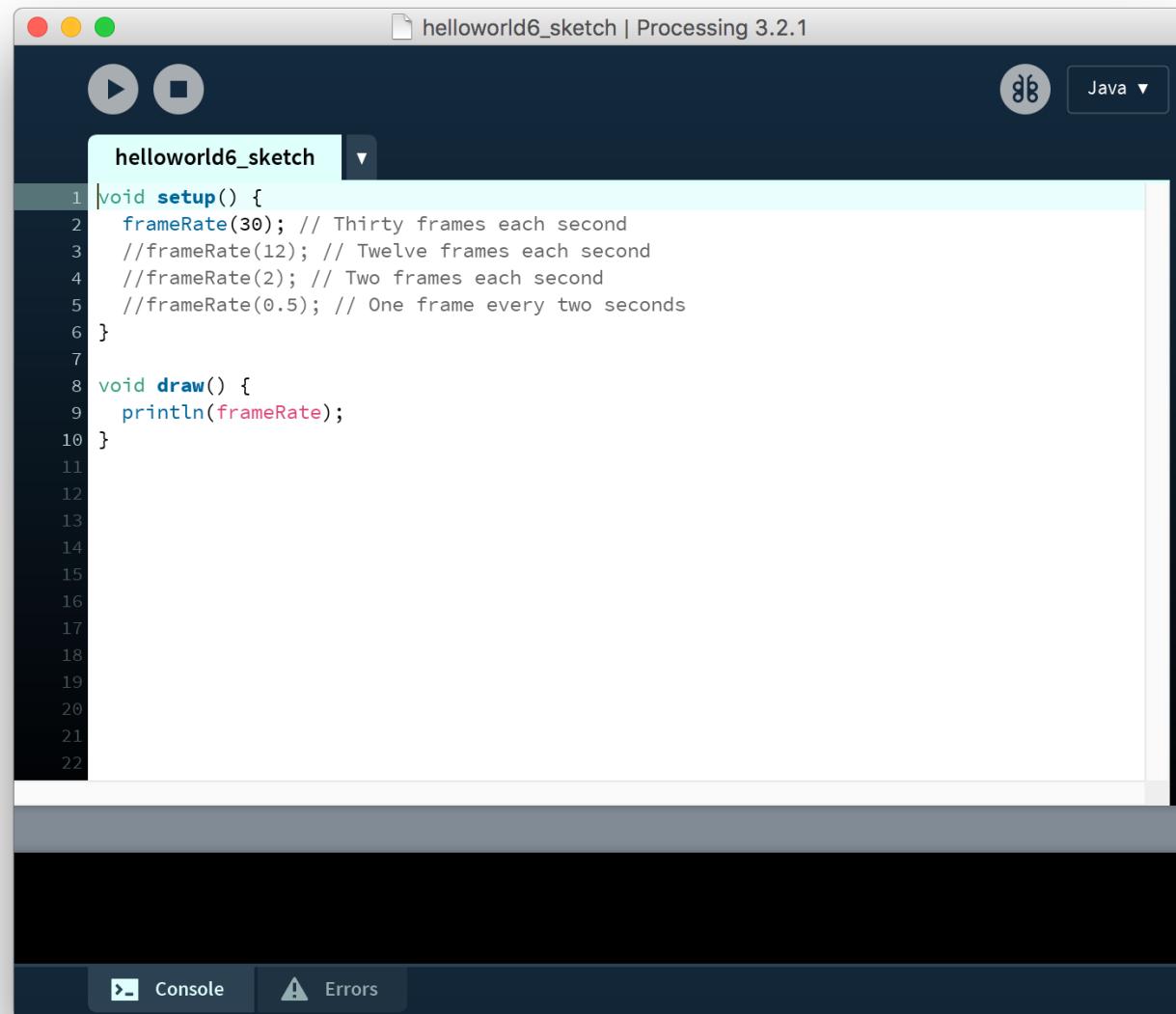
```
1 background(255) ;
2 size(800, 800) ;
3
4 // Examples of loading a text file and a JPEG image
5 // from the data folder of a sketch.
6
7
8 String[] lines = loadStrings("helloworld.txt");
9 PImage image = loadImage("gumball.jpg");
10
11 println( "Image Width: " + image.height ) ;
12 println( "Image Height: " + image.width ) ;
13 printArray( lines ) ;
14
15 // Displays the image (centered on canvas) at its actual size
16 image(image, (width-image.width)/2, (height-image.height)/2);
17
18
19
20
```

Image Width: 553
Image Height: 570
[0] "Hello World!"
[1] "Processing is Cool!"

Console Errors



Frame Rates



The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld6_sketch | Processing 3.2.1". The main window displays the following Java code:

```
1 void setup() {
2     frameRate(30); // Thirty frames each second
3     //frameRate(12); // Twelve frames each second
4     //frameRate(2); // Two frames each second
5     //frameRate(0.5); // One frame every two seconds
6 }
7
8 void draw() {
9     println(frameRate);
10}
11
12
13
14
15
16
17
18
19
20
21
22
```

The code defines a `setup()` function that sets the frame rate to 30 frames per second. It also contains a `draw()` function that prints the current frame rate to the console. The Processing logo and Java dropdown menu are visible in the top right corner. At the bottom, there are tabs for "Console" and "Errors".

helloworld6_sketch | Processing 3.2.1

Java ▾

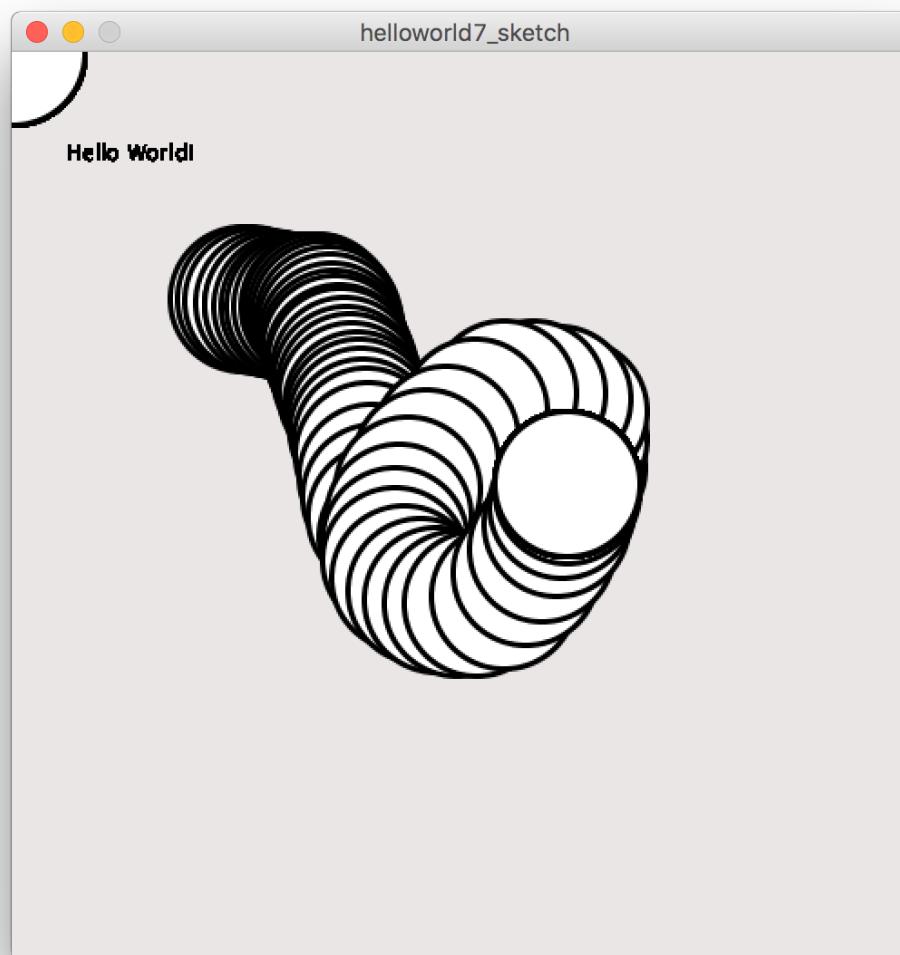
helloworld6_sketch ▾

```
1 void setup() {
2     frameRate(30); // Thirty frames each second
3     //frameRate(12); // Twelve frames each second
4     //frameRate(2); // Two frames each second
5     //frameRate(0.5); // One frame every two seconds
6 }
7
8 void draw() {
9     println(frameRate);
10}
11
12
```

30.712574
30.599195
30.364637
30.108313
30.17732
30.461428
30.461624
30.054953
29.996319
30.023298
30.04973
30.30662
30.062653

Console Errors

Hello World with Java Objects



The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld7_sketch | Processing 3.2.1". The top menu bar includes "File", "Edit", "Select", "Sketch", "Tools", "Help", and "About". On the right side of the top bar are icons for "Run", "Stop", "Reset", "Debug", and "Java". The main workspace displays the code for the sketch "helloworld7_sketch". The code uses the HelloObject class:

```
1 |
2 int x = 30;
3 HelloObject obj = new HelloObject() ;
4
5 void setup()
6 {
7     size(500, 500);
8     smooth() ;
9     strokeWeight(3);
10    strokeCap(ROUND);
11    background(#EAE6E6);
12
13    // Only draw once
14    //noLoop();
15 }
16
17 void draw() {
18
19     //background(#EAE6E6);
20
21     // print hello world
22     fill(0);
```

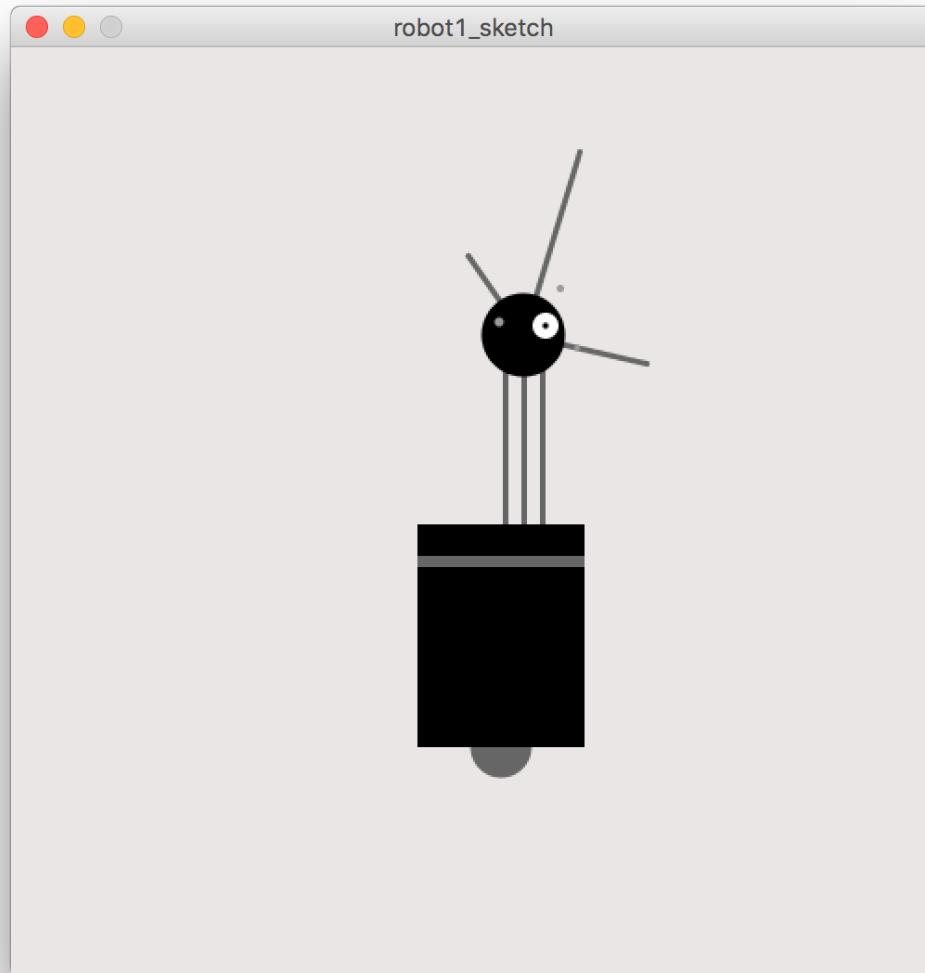
The bottom navigation bar features tabs for "Console" and "Errors".

The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "helloworld7_sketch | Processing 3.2.1". The top menu bar includes standard Mac OS X icons (red, yellow, green) and a toolbar with play/pause and stop buttons. On the right, there's a circular icon with a 'gg' logo and a "Java ▾" dropdown. The main workspace has tabs for "helloworld7_sketch" (selected) and "HelloObject". The code editor displays the following Java code:

```
1
2 class HelloObject
3 {
4     String getMessage() { return "Hello World!" ; }
5 }
```

The code editor has line numbers from 1 to 22 on the left. Below the code editor is a large gray status bar. At the bottom, there are two tabs: "Console" with a terminal icon and "Errors" with an exclamation mark icon.

Hello HAL (the Robot)



robot1_sketch | Processing 3.2.1

Robot

```
1
2 int x = 30;
3 Robot hal = new Robot() ;
4
5 void setup()
6 {
7     size(500, 500);
8     smooth() ;
9     strokeWeight(3);
10    strokeCap(ROUND);
11    background(#EAE6E6);
12
13    // Only draw once
14    noLoop();
15 }
16
17 void draw() {
18     hal.draw() ;
19 }
```

Console Errors

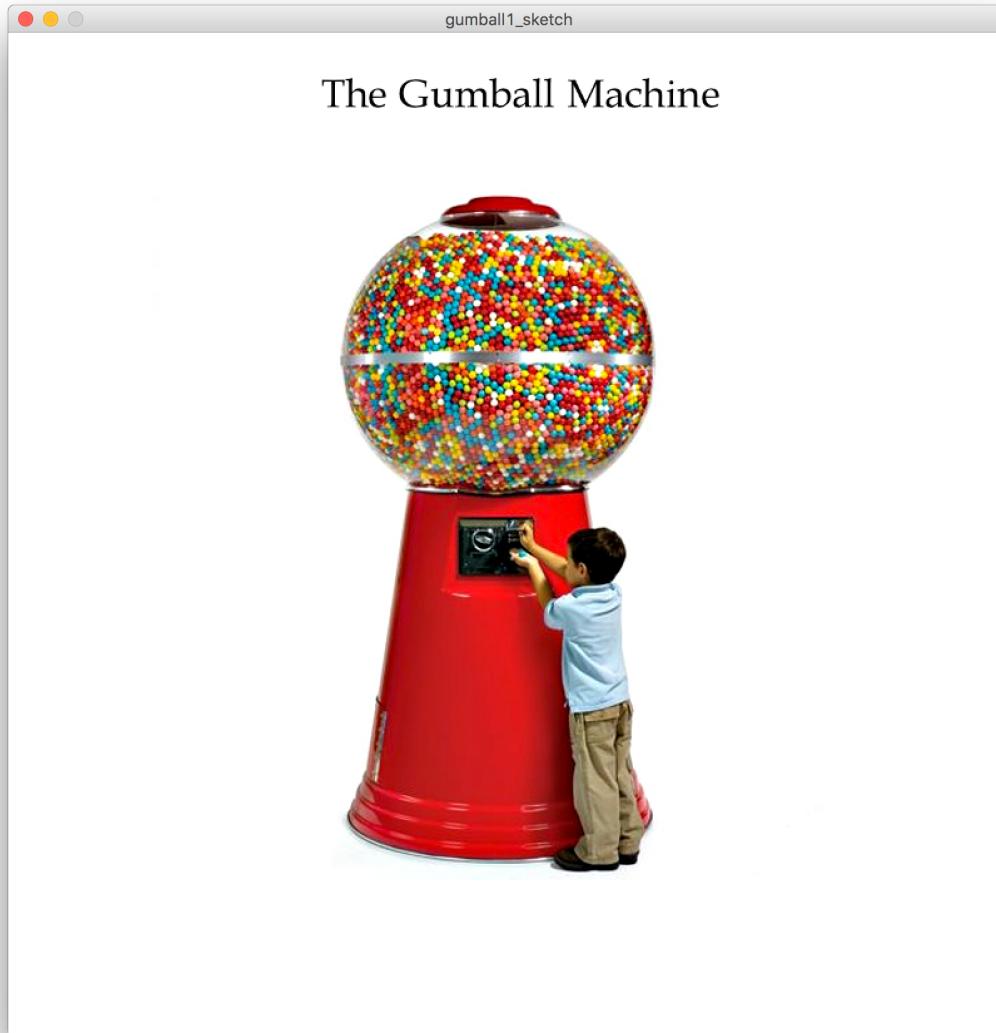
robot1_sketch | Processing 3.2.1

Robot

```
1 class Robot
2 {
3
4     public void draw()
5     {
6         //size(720, 480);
7         //smooth();
8         //strokeWeight(2);
9         //background(204);
10        //ellipseMode(RADIUS);
11
12        // Neck
13        stroke(102);           // Set stroke to gray
14        line(266, 257, 266, 162); // Left
15        line(276, 257, 276, 162); // Middle
16        line(286, 257, 286, 162); // Right
17
18        // Antennae
19        line(276, 155, 246, 112); // Small
20        line(276, 155, 306, 56); // Tall
21        line(276, 155, 342, 170); // Medium
22    }
}
```

Console Errors

Gumball Machine (v1)



The screenshot shows the Processing 3.2.1 IDE interface. The title bar reads "gumball1_sketch | Processing 3.2.1". The top menu bar includes standard OS X window controls (red, yellow, green) and tabs for "gumball1_sketch", "GumballMachine", "State", and a dropdown menu. Below the tabs is a toolbar with a play button, a square button, a circular icon with "gg", and a "Java" dropdown. The main area displays the following Java code:

```
1
2
3 void setup()
4 {
5     size(800, 800) ;
6     background(255) ;
7     smooth() ;
8     strokeWeight(3) ;
9     strokeCap(ROUND) ;
10
11    // load font
12    PFont font;
13    font = loadFont("BookAntiqua-48.vlw");
14    textAlign(CENTER);
15
16    // Only draw once
17    noLoop();
18 }
19
20 void draw() {
21     fill(0);
22     text("The Gumball Machine", 250, 60);
23     PImage image = loadImage("gumball.jpg");
24     image(image, (width-image.width)/2, (height-image.height)/2);
25     runTest() ;
26 }
27
28
29 public void runTest() {
```

The code sets up a 800x800 pixel canvas with a white background, smooth rendering, and a stroke weight of 3 pixels. It loads a font from "BookAntiqua-48.vlw" and sets the text alignment to center. The draw() function fills the entire canvas with black, displays the text "The Gumball Machine" at position (250, 60), loads an image named "gumball.jpg", and displays it centered on the canvas. The runTest() function is currently empty.

gumball1_sketch | Processing 3.2.1

Java ▾

gumball1_sketch GumballMachine State ▾

```
1 public class GumballMachine {  
2     State soldOutState;  
3     State noQuarterState;  
4     State hasQuarterState;  
5     State soldState;  
6  
7     State state = soldOutState;  
8     int count = 0;  
9  
10    public GumballMachine(int numberGumballs) {  
11        soldOutState = new SoldOutState(this);  
12        noQuarterState = new NoQuarterState(this);  
13        hasQuarterState = new HasQuarterState(this);  
14        soldState = new SoldState(this);  
15  
16        this.count = numberGumballs;  
17        if (numberGumballs > 0) {  
18            state = noQuarterState;  
19        }  
20    }  
21  
22}
```

Console Errors

gumball1_sketch | Processing 3.2.1

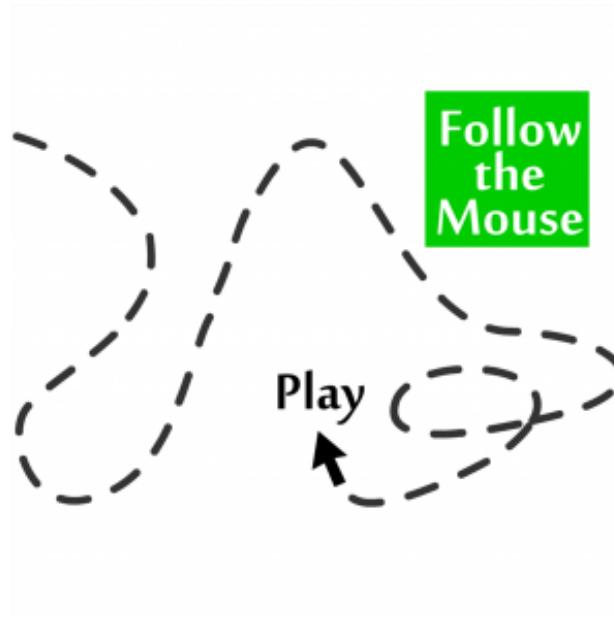
Java ▾

gumball1_sketch GumballMachine State ▾

```
1 public interface State {  
2     public void insertQuarter();  
3     public void ejectQuarter();  
4     public void turnCrank();  
5     public void dispense();  
6 }  
7  
8  
9  
10  
11    public class HasQuarterState implements State {  
12        GumballMachine gumballMachine;  
13  
14        public HasQuarterState(GumballMachine gumballMachine) {  
15            this.gumballMachine = gumballMachine;  
16        }  
17  
18        public void insertQuarter() {  
19            System.out.println("You can't insert another quarter");  
20        }  
21  
22        public void ejectQuarter() {
```

Console Errors

Follow Me



follow1_sketch | Processing 3.2.1

Java ▾

follow1_sketch

```
1 void setup() {  
2     size(480, 120);  
3     fill(0, 102);  
4     noStroke();  
5 }  
6  
7 void draw() {  
8     ellipse(mouseX, mouseY, 9, 9);  
9 }  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22
```

Done saving.

follow1_sketch

The image shows a Processing sketch window titled "follow1_sketch". Inside the sketch, a mouse cursor is visible, and a trail of gray dots follows its path. The sketch window has a title bar with three buttons (red, yellow, green) and a close button. Below the title bar is a toolbar with a play button, a square button, and a Java dropdown menu. The main area contains the Processing code. At the bottom of the sketch window, there is a message "Done saving." and a scroll bar. The entire window is set against a dark background.

Console Errors

follow2_sketch | Processing 3.2.1

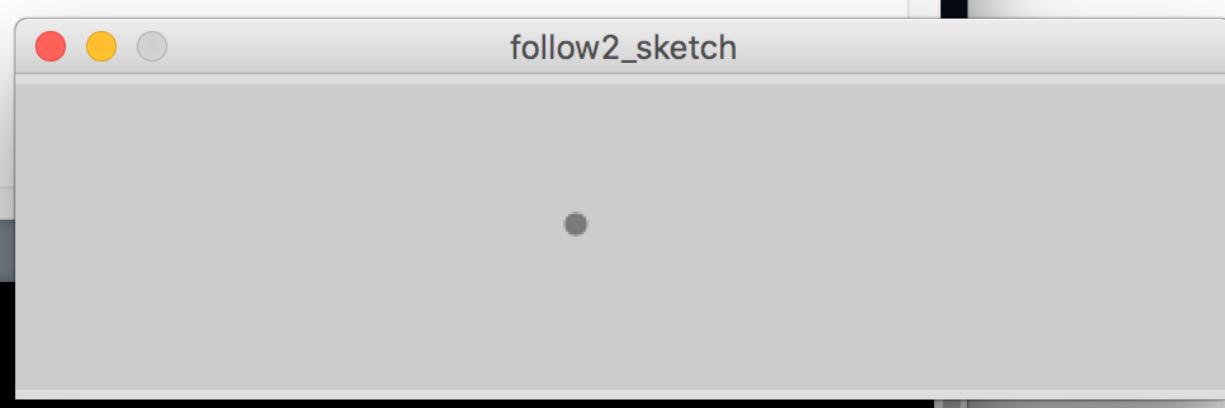
Java ▾

follow2_sketch

```
1 void setup() {  
2     size(480, 120);  
3     fill(0, 102);  
4     noStroke();  
5 }  
6  
7 void draw() {  
8     background(204);  
9     ellipse(mouseX, mouseY, 9, 9);  
10 }  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22
```

follow2_sketch

Console Errors



follow3_sketch | Processing 3.2.1

Java ▾

```
follow3_sketch
```

```
1 void setup() {  
2     size(480, 120);  
3     strokeWeight(4);  
4     stroke(0, 102);  
5 }  
6  
7 void draw() {  
8     line(mouseX, mouseY, pmouseX, pmouseY);  
9 }
```

Done saving.

follow3_sketch

Done saving.

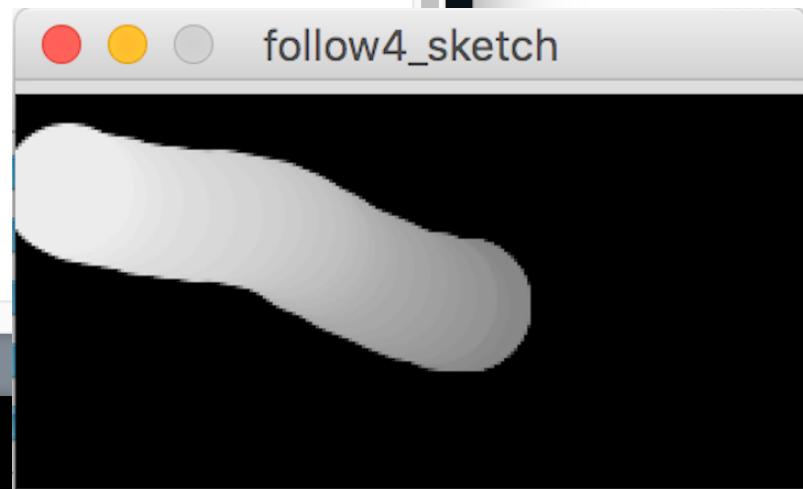
Console Errors

follow4_sketch | Processing 3.2.1

Java ▾

```
1 int num = 60;
2 int[] x = new int[num];
3 int[] y = new int[num];
4
5 void setup() {
6     size(240, 120);
7     noStroke();
8 }
9
10 void draw() {
11     background(0);
12     // Copy array values from back to front
13     for (int i = x.length-1; i > 0; i--) {
14         x[i] = x[i-1];
15         y[i] = y[i-1];
16     }
17     x[0] = mouseX; // Set the first element
18     y[0] = mouseY; // Set the first element
19     for (int i = 0; i < x.length; i++) {
20         fill(i * 4);
21         ellipse(x[i], y[i], 40, 40);
22     }
}
```

Console Errors



Keyboard Events



keypress1_sketch | Processing 3.2.1

Java ▾

```
keypress1_sketch
1 void setup() {
2     size(120, 120);
3     textSize(64);
4     textAlign(CENTER);
5 }
6
7 void draw() {
8     background(0);
9     text(key, 60, 80);
10}
11
12
13
14
15
16
17
18
19
20
21
22
```

The Processing sketch window displays the output of the provided code. A large white lowercase letter 'a' is centered on a black background within a white frame. The window has a title bar 'keypress1_sketch | Processing 3.2.1' and a status bar at the bottom with tabs for 'Console' and 'Errors'.

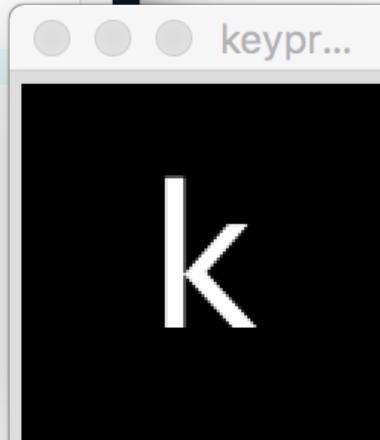
keypress2_sketch | Processing 3.2.1

Java ▾

keypress2_sketch

```
1 void setup() {  
2     size(120, 120);  
3     textSize(64);  
4     textAlign(CENTER);  
5 }  
6  
7 void draw() {  
8 }  
9  
10 void mousePressed() {  
11 }  
12  
13 void keyPressed() {  
14     background(0);  
15     text(key, 60, 80);  
16 }  
17  
18  
19  
20  
21  
22
```

keypr...



Console Errors

keypress3_sketch | Processing 3.2.1

Java ▾

keypress3_sketch

```
1 int x = 215;
2
3 void setup() {
4     size(480, 120);
5 }
6
7 void draw() {
8     if (keyPressed && (key == CODED)) { // If it's a coded key
9         if (keyCode == LEFT) { // If it's the left arrow
10            x--;
11        } else if (keyCode == RIGHT) { // If it's the right arrow
12            x++;
13        }
14    }
15    rect(x, 45, 50, 50);
16 }
17
18
19
20
21
22
```

keypress3_sketch

Console Errors

keypress4_sketch | Processing 3.2.1

Java ▾

```
1 void setup() {
2     size(200, 200) ;
3     background(255) ;
4 }
5
6
7 void draw() []
8 }
9
10 void mousePressed() {
11     stroke(0) ;
12     fill(175) ;
13     rectMode(CENTER) ;
14     rect(mouseX, mouseY, 16, 16) ;
15 }
16
17 void keyPressed() {
18     background(255) ;
19 }
```

Done saving.

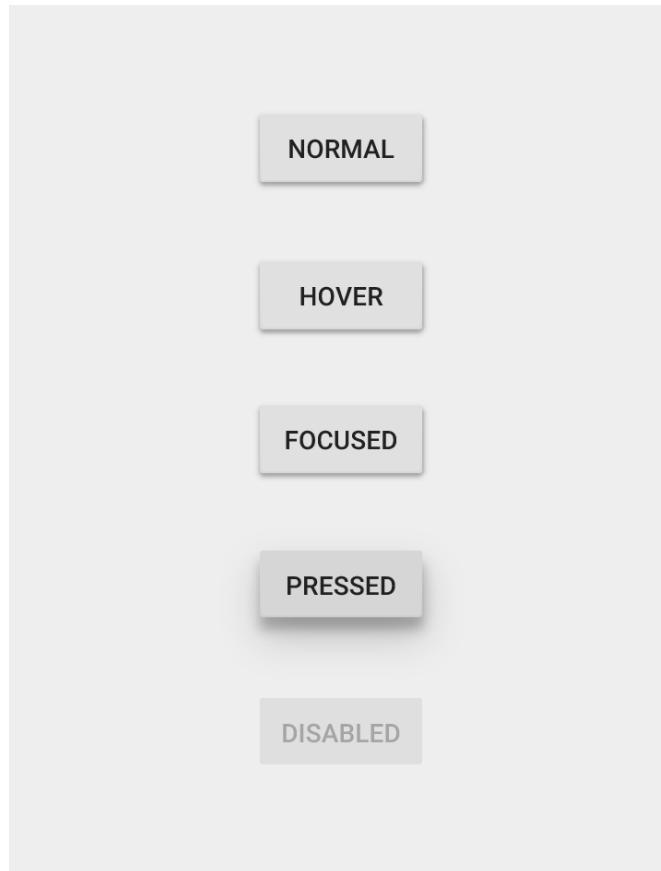
Console Errors

The image shows a Processing sketch titled "keypress4_sketch" running in the Processing 3.2.1 IDE. The sketch consists of a main code editor and a preview window. The code editor contains the following Pseudocode:

```
1 void setup() {
2     size(200, 200) ;
3     background(255) ;
4 }
5
6
7 void draw() []
8 }
9
10 void mousePressed() {
11     stroke(0) ;
12     fill(175) ;
13     rectMode(CENTER) ;
14     rect(mouseX, mouseY, 16, 16) ;
15 }
16
17 void keyPressed() {
18     background(255) ;
19 }
```

The preview window displays a white canvas with a 4x3 grid of 10 small gray squares. The squares are positioned at various coordinates, with some appearing multiple times. The top row has two squares at approximately (100, 100) and (100, 180). The second row has three squares at approximately (150, 100), (150, 180), and (200, 100). The third row has four squares at approximately (100, 150), (150, 150), (150, 200), and (200, 150). The bottom row has one square at approximately (100, 200).

Making UI Buttons



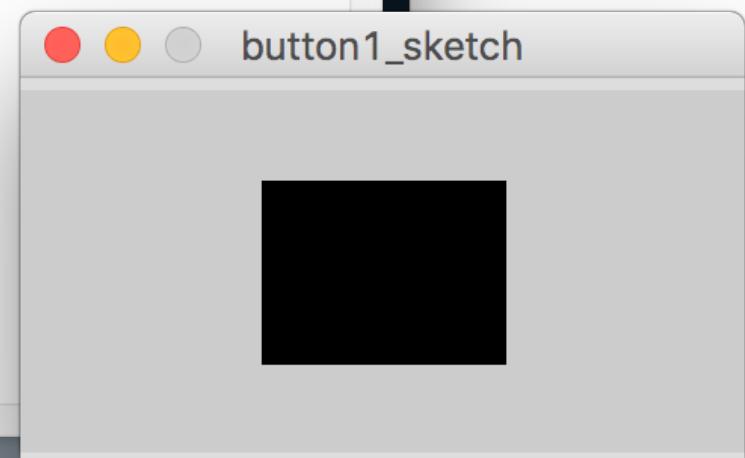
button1_sketch | Processing 3.2.1

Java ▾

button1_sketch

```
1 int x = 80;
2 int y = 30;
3 int w = 80;
4 int h = 60;
5
6 void setup() {
7     size(240, 120);
8 }
9
10 void draw() {
11     background(204);
12     if ((mouseX > x) && (mouseX < x+w) &&
13         (mouseY > y) && (mouseY < y+h)) {
14         fill(0);
15     } else {
16         fill(255);
17     }
18     rect(x, y, w, h);
19 }
20
21
22 }
```

Console Errors



button2_sketch | Processing 3.2.1

Java ▾

button2_sketch

```
1 void setup() {
2     size(240, 120);
3     strokeWeight(30);
4 }
5
6 void draw() {
7     background(204);
8     stroke(102);
9     line(40, 0, 70, height);
10    if (mousePressed == true) {
11        stroke(0);
12    }
13    else {
14        stroke(255);
15    }
16    line(0, 70, width, 50);
17 }
18
19
20
21
22
```

Done saving.

button2_sketch

Console Errors

button3_sketch | Processing 3.2.1

Java ▾

```
button3_sketch
1 void setup() {
2     size(120, 120);
3     strokeWeight(30);
4 }
5
6 void draw() {
7     background(204);
8     stroke(102);
9     line(40, 0, 70, height);
10    if (mousePressed) {
11        if (mouseButton == LEFT) {
12            stroke(255);
13        } else {
14            stroke(0);
15        }
16        line(0, 70, width, 50);
17    }
18 }
19
20
21
22 }
```

button...

Console Errors

Pacman



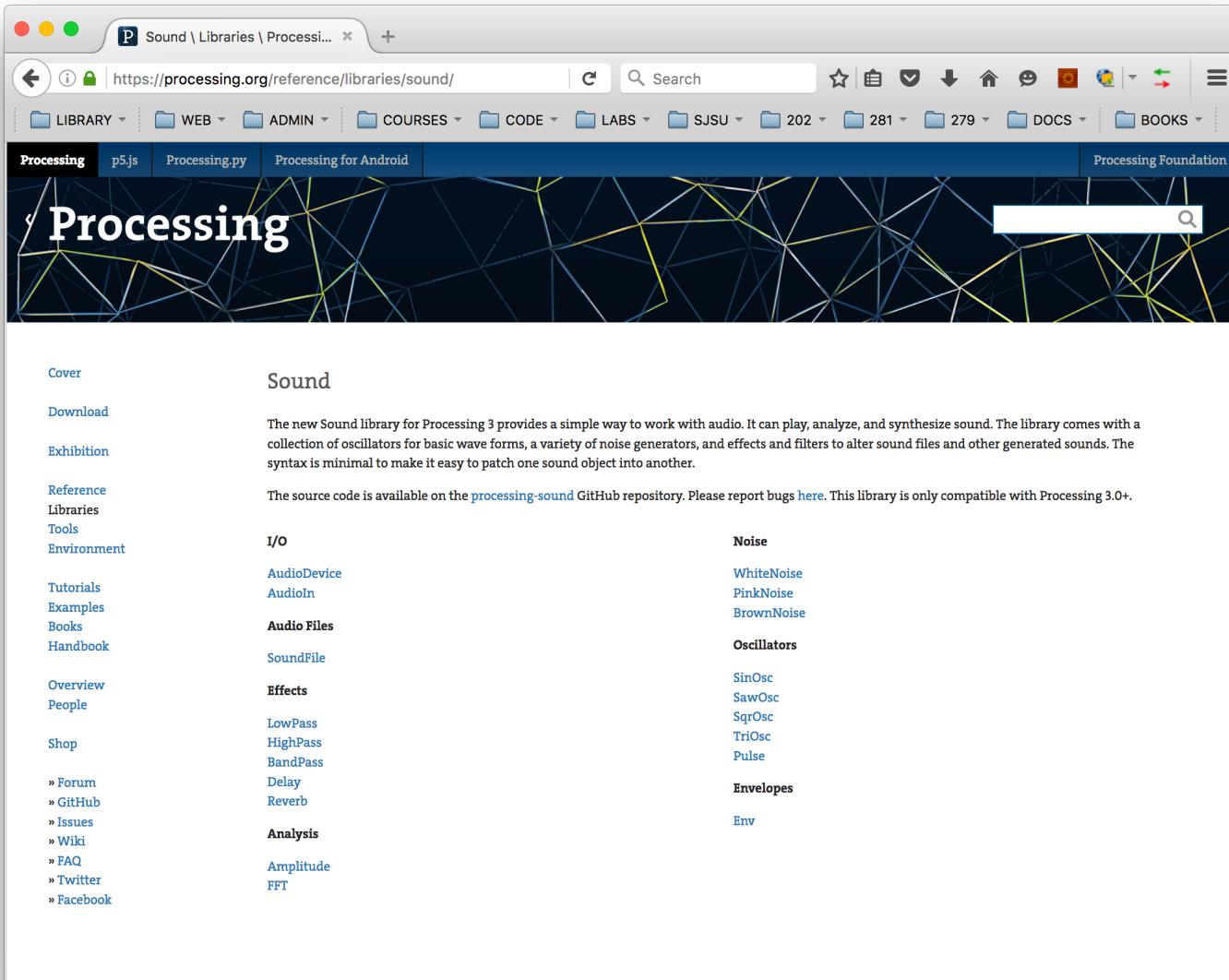
The screenshot shows the Processing 3.2.1 IDE interface. At the top, there are three red, yellow, and green window control buttons. The title bar displays "pacman_sketch | Processing 3.2.1". Below the title bar are two large circular buttons: a play button on the left and a stop button on the right. To the right of these buttons is a circular icon containing a double-headed arrow symbol, and next to it is a dropdown menu labeled "Java ▾".

The main area of the window contains the code for the sketch:

```
pacman_sketch
1 int radius = 40;
2 float x = 110;
3 float speed = 0.5;
4 int direction = 1;
5
6 void setup() {
7     size(240, 120);
8     ellipseMode(RADIUS);
9 }
10
11 void draw() {
12     background(0);
13     x += speed * direction;
14     if ((x > width-radius) || (x < radius)) {
15         direction = -direction; // Flip direction
16     }
17     if (direction == 1) {
18         arc(x, 60, radius, radius, 0.52, 5.76); // Face right
19     } else {
20         arc(x, 60, radius, radius, 3.67, 8.9); // Face left
21     }
22 }
```

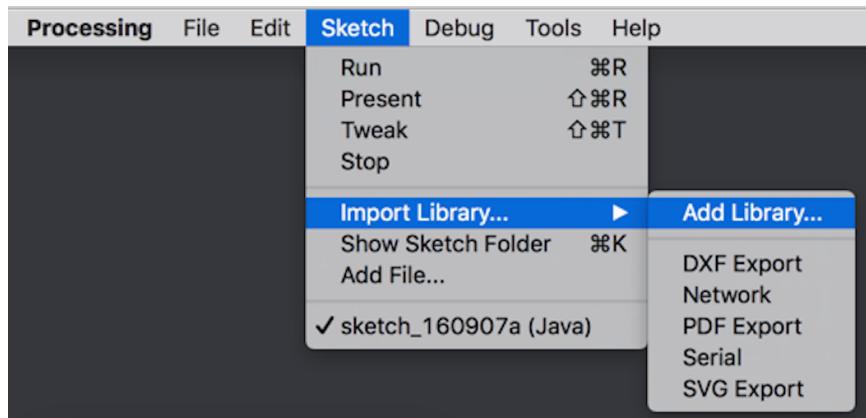
At the bottom of the window, there are two tabs: "Console" and "Errors". The "Console" tab is active, indicated by its bolded text and a small play icon. The "Errors" tab has a warning icon.

Playing Sound



The screenshot shows a web browser window displaying the Processing.org reference library for the Sound library. The URL in the address bar is <https://processing.org/reference/libraries/sound/>. The page features a dark background with a geometric, wireframe-like pattern. At the top, there's a navigation bar with links to various sections like LIBRARY, WEB, ADMIN, COURSES, CODE, LABS, SJSU, etc. Below the navigation bar is a blue header bar with tabs for Processing, p5.js, Processing.py, Processing for Android, and Processing Foundation. A search bar is located in the top right corner. The main content area has a large "Processing" logo on the left. To the right of the logo, there's a search bar with a magnifying glass icon. The page content is organized into two columns. The left column contains links for Cover, Download, Exhibition, Reference, Libraries, Tools, Environment, Tutorials, Examples, Books, Handbook, Overview, People, Shop, and social media links for Forum, GitHub, Issues, Wiki, FAQ, Twitter, and Facebook. The right column contains links for Sound, I/O, Effects, Analysis, Noise, Oscillators, and Envelopes. Each category lists specific classes or functions, such as AudioDevice, AudioIn, SoundFile, LowPass, HighPass, BandPass, Delay, Reverb, Amplitude, FFT, WhiteNoise, PinkNoise, BrownNoise, SinOsc, SawOsc, SqrOsc, TriOsc, Pulse, Env, and Env.

Sound Library Not Installed by Default!



A screenshot of the Contribution Manager window. The 'Libraries' tab is selected. A search bar at the top right shows 'Sound'. A list of libraries is shown, with 'Sound | Sound library for Processing.' highlighted. At the bottom, there is a detailed view for the Sound library, showing its version (1.3.2), author ('The Processing Foundation'), description ('Sound library for Processing.'), and installation options: 'Install', 'Update', and 'Remove'.

sound1_sketch | Processing 3.2.1

Java ▾

```
sound1_sketch
import processing.sound.*;
SoundFile blip;
int radius = 120;
float x = 0;
float speed = 1.0;
int direction = 1;

void setup() {
    size(440, 440);
    ellipseMode(RADIUS);
    blip = new SoundFile(this, "blip.wav");
    x = width/2; // Start in the center
}

void draw() {
    background(0);
    x += speed * direction;
    if ((x > width-radius) || (x < radius)) {
        direction = -direction; // Flip direction
        blip.play();
    }
}
```

sound1_sketch

Console Errors

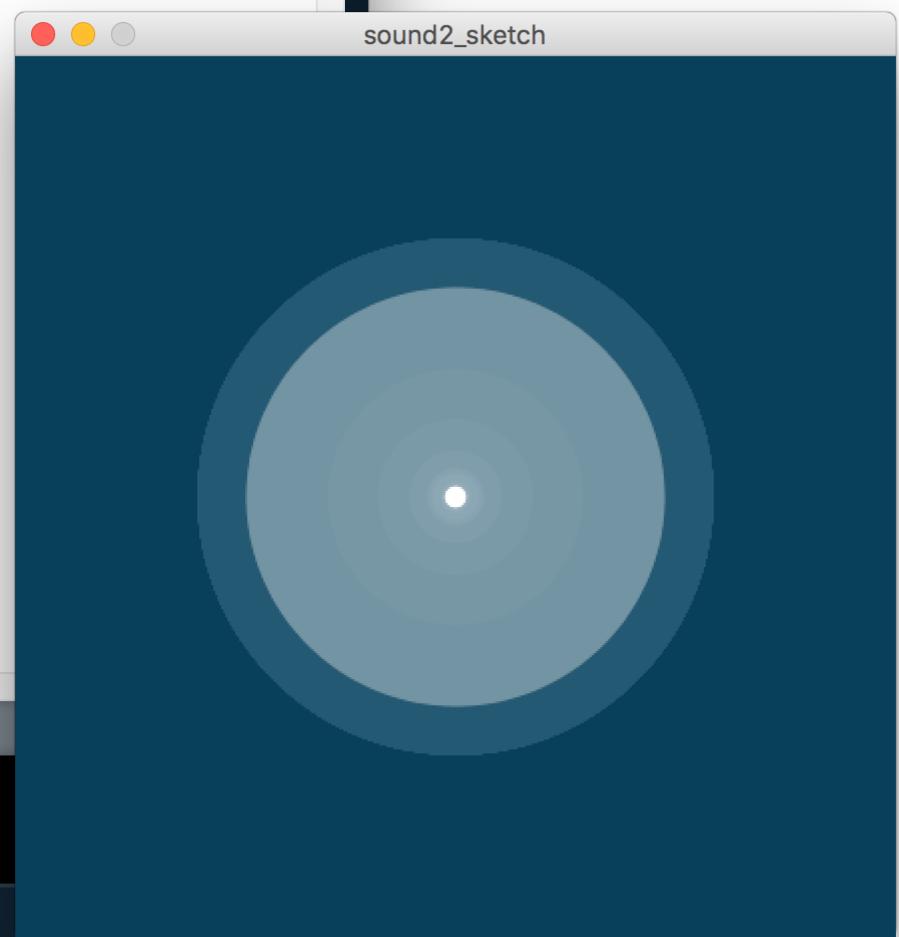
sound2_sketch | Processing 3.2.1

sound2_sketch

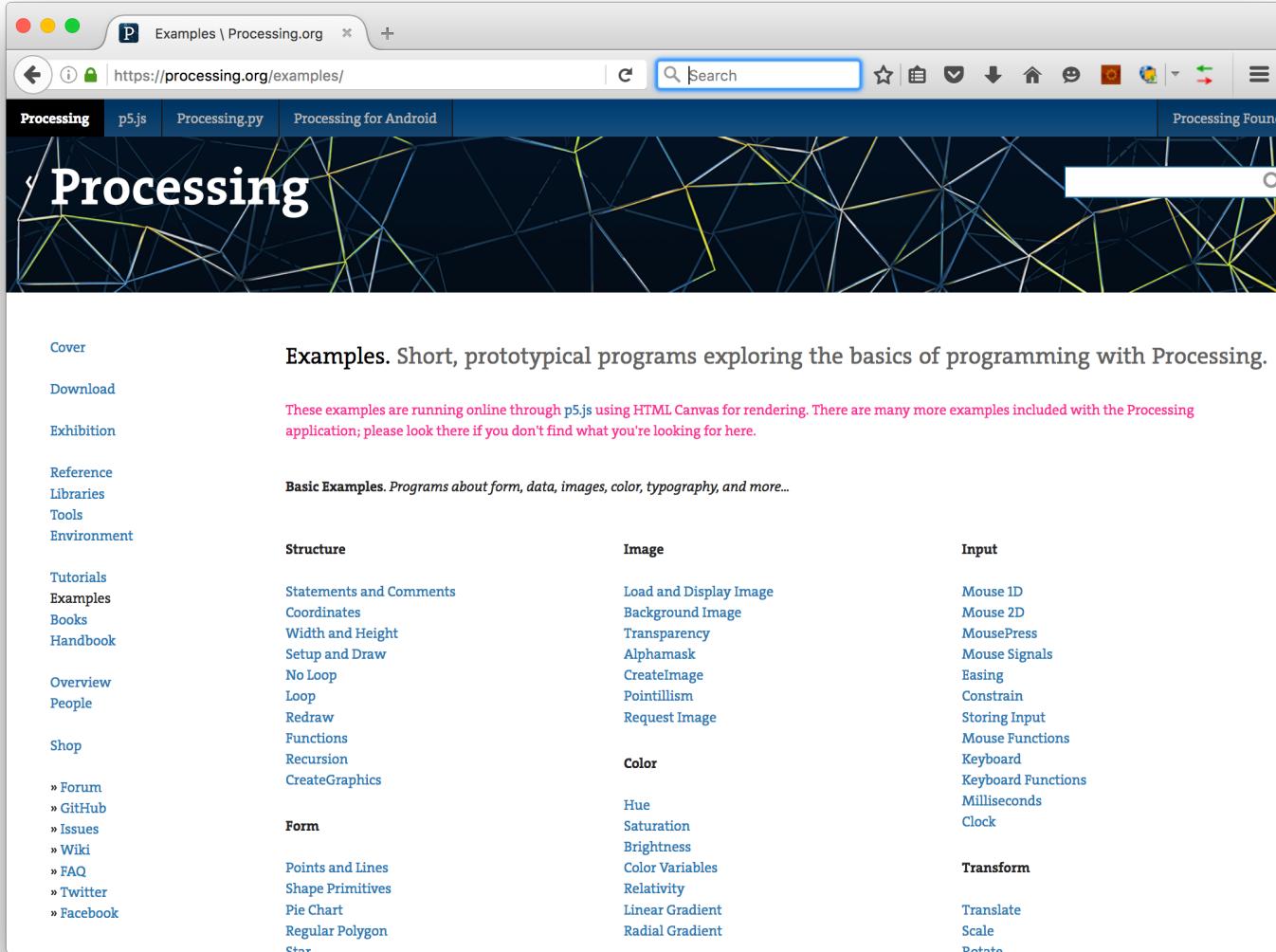
```
1 import processing.sound.*;
2
3 AudioIn mic;
4 Amplitude amp;
5
6 void setup() {
7     size(440, 440);
8     background(0);
9     // Create an audio input and start it
10    mic = new AudioIn(this, 0);
11    mic.start();
12    // Create a new amplitude analyzer and patch into input
13    amp = new Amplitude(this);
14    amp.input(mic);
15 }
16
17 void draw() {
18     // Draw a background that fades to black
19     noStroke();
20     fill(26, 76, 102, 10);
21     rect(0, 0, width, height);
22     // The analyze() method returns values between 0 and 1,
23     // so map() is used to convert the values to larger numbers
24     float diameter = map(amp.analyze(), 0, 1, 10, width);
25     // Draw the circle based on the volume
26     fill(255);
27     ellipse(width/2, height/2, diameter, diameter);
28 }
29
```

Java ▾

Console Errors



More Examples

A screenshot of a web browser displaying the 'Examples' section of the Processing.org website. The URL in the address bar is https://processing.org/examples/. The page features a large banner image of a complex geometric pattern of blue and yellow lines on a dark background. The main content area has a light gray background. On the left, there's a sidebar with links to 'Cover', 'Download', 'Exhibition', 'Reference', 'Libraries', 'Tools', 'Environment', 'Tutorials', 'Examples', 'Books', 'Handbook', 'Overview', 'People', 'Shop', and social media links for '» Forum', '» GitHub', '» Issues', '» Wiki', '» FAQ', '» Twitter', and '» Facebook'. The main content area contains three columns: 'Structure' (Statements and Comments, Coordinates, Width and Height, Setup and Draw, No Loop, Loop, Redraw, Functions, Recursion, CreateGraphics), 'Image' (Load and Display Image, Background Image, Transparency, Alphamask, CreateImage, Pointillism, Request Image), 'Input' (Mouse 1D, Mouse 2D, MousePress, Mouse Signals, Easing, Constrain, Storing Input, Mouse Functions, Keyboard, Keyboard Functions, Milliseconds, Clock), 'Color' (Hue, Saturation, Brightness, Color Variables, Relativity), and 'Transform' (Linear Gradient, Radial Gradient, Translate, Scale, Rotate).

Cover

Download

Exhibition

Reference

Libraries

Tools

Environment

Tutorials

Examples

Books

Handbook

Overview

People

Shop

» Forum

» GitHub

» Issues

» Wiki

» FAQ

» Twitter

» Facebook

Structure

Statements and Comments

Coordinates

Width and Height

Setup and Draw

No Loop

Loop

Redraw

Functions

Recursion

CreateGraphics

Image

Load and Display Image

Background Image

Transparency

Alphamask

CreateImage

Pointillism

Request Image

Input

Mouse 1D

Mouse 2D

MousePress

Mouse Signals

Easing

Constrain

Storing Input

Mouse Functions

Keyboard

Keyboard Functions

Milliseconds

Clock

Color

Hue

Saturation

Brightness

Color Variables

Relativity

Transform

Linear Gradient

Radial Gradient

Translate

Scale

Rotate

<https://processing.org/examples/>

Games- Collections at OpenProcessing.org Examples | Processing.org Libraries | Processing.org

https://processing.org/examples/ games processing

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

Processing p5.js Processing.py Processing for Android Processing Foundation

Processing

Cover Examples. Short, prototypical programs exploring the basics of programming with Processing.

Download These examples are running online through p5.js using HTML Canvas for rendering. There are many more examples included with the Processing application; please look there if you don't find what you're looking for here.

Exhibition

Reference Basic Examples. Programs about form, data, images, color, typography, and more...

Libraries

Tools

Environment

Tutorials

Examples

Books

Handbook

Overview

People

Shop

» Forum

» GitHub

» Issues

» Wiki

» FAQ

» Twitter

» Facebook

Structure	Image	Input
Statements and Comments	Load and Display Image	Mouse 1D
Coordinates	Background Image	Mouse 2D
Width and Height	Transparency	MousePress
Setup and Draw	Alphamask	Mouse Signals
No Loop	CreateImage	Easing
Loop	Pointillism	Constrain
Redraw	Request Image	Storing Input
Functions		Mouse Functions
Recursion		Keyboard
CreateGraphics		Keyboard Functions
	Color	Milliseconds
	Hue	Clock
	Saturation	
	Brightness	
	Color Variables	Transform
	Relativity	
	Linear Gradient	Translate
	Radial Gradient	Scale

font Highlight All Match Case 1 of 4 matches

Games- Collections at OpenProcessing.org Examples | Processing.org Libraries | Processing.org

www.openprocessing.org/collection/25 games processing

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

OpenProcessing

browse classrooms collections books go plus+ jobs search

login - sign up

sketches from Games

This collection includes games that are built with Processing. Enjoy! (and feel free to submit sketches that I missed)

collection created by [Sinan Ascioglu](#)
includes sketches by [Art Merrideth](#) (4) [Jarod Searle](#) (1)
[oggy](#) (2) [Yuri Socher Bichibichi](#) (1)
[Greg Wittmann](#) (3) [Tetsuya Matsuno](#) (1)
[Lambert Wang](#) (4) [MacKenzie Bates](#) (1)
[Mark Gillespie](#) (1) [Juan Irache](#) (1)
[jacques maire](#) (1) [Robert D.](#) (1)
[Ricardo Gambirasio](#) (1) [N. Strang](#) (1)
[haroflow](#) (1) [adrian trevino](#) (1)

submit a sketch from your portfolio

Select a Sketch

or you can enter the visualID of the sketch below
<http://openprocessing.org/sketch/> Go

Share Subscribe to RSS

Font Highlight All Match Case Phrase not found

The screenshot shows a web browser window with three tabs open:

- Games- Collections at OpenProcessing.org
- Examples | Processing.org
- Libraries | Processing.org

The main content area displays the "Libraries" section of the Processing.org website. The header features the Processing logo and navigation links for "LIBRARY", "WEB", "ADMIN", "CODE", "EDU", "LABS", "SJSU", "202", "281", "279", "DOCS", "ZAPP", and "BOOKS".

The main content includes:

- A large banner image with the word "Processing" overlaid.
- A search bar.
- Links for "Cover", "Download", and "Exhibition".
- A section titled "Libraries. Extend Processing beyond graphics and images into audio, video, and communication with other devices.".
- Information about libraries created by the Processing Foundation, mentioning PDF Export, Network, Serial, and DXF Export.
- Descriptions for various libraries:
 - PDF Export**: Create PDF files. These vector graphics files can be scaled to any size and printed at high resolutions.
 - Network**: Send and receive data over the Internet through simple clients and servers.
 - Serial**: Send data between Processing and external hardware through serial communication (RS-232).
 - DXF Export**: Create DXF files to save geometry for loading into other programs. It works with triangle-based graphics including polygons, boxes, and spheres.
 - Video**: Read images from a camera, play movie files, and create movies.
 - Sound**: Playback audio files, audio input, synthesize sound, and effects.
 - Hardware I/O**: Access peripherals on the Raspberry Pi and other Linux-based computers.
- A "Contributions" section explaining how to download contributed libraries.
- Footnotes providing additional information about contributed libraries and their development.

Games- Collections at OpenProcessing.org Examples | Processing.org Books | Processing.org

https://processing.org/books/ games processing

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

Processing p5.js Processing.py Processing for Android Processing Foundation

Processing

Cover Download Exhibition Reference Libraries Tools Environment Tutorials Examples Books Handbook Overview People Shop » Forum » GitHub » Issues » Wiki » FAQ » Twitter » Facebook

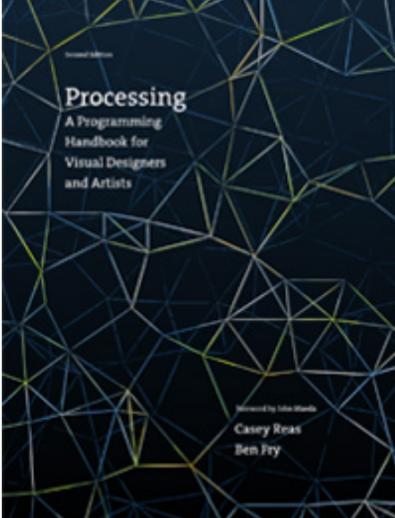
Books. Processing books cover topics from programming basics to visualization. Browse this page to find the right books for you.

Processing: A Programming Handbook for Visual Designers, Second Edition
Casey Reas and Ben Fry.
Published December 2014, The MIT Press. 720 pages. Hardcover.
[» Order from MIT Press](#)
[» Order from Amazon](#)

The second edition of the Handbook has been thoroughly updated, influenced by the seven years of Processing being taught in classrooms, computer labs, and studios since the first edition. Every chapter has been revised, and added chapters introduce new ways to work with data and geometry. New "synthesis" chapters offer discussion and worked examples of such topics as sketching with code, modularity, and algorithms. Interviews have been added that cover a wider range of projects and history. "Extension" chapters are now offered online so they can be updated to keep pace with technological developments in such fields as computer vision and electronics.

If you are an educator, you can request a [desk/exam copy](#) from the MIT Press website.

Make: Getting Started with Processing, Second Edition
Casey Reas and Ben Fry.
Published September 2015, Maker Media. 238 pages. Paperback.



Getting Started with Proce... Contact Us: Step 1 of 2 | T... LP Learning Processing 2nd E... +

shop.oreilly.com/product/0636920031406.do Search

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

 O'REILLY®

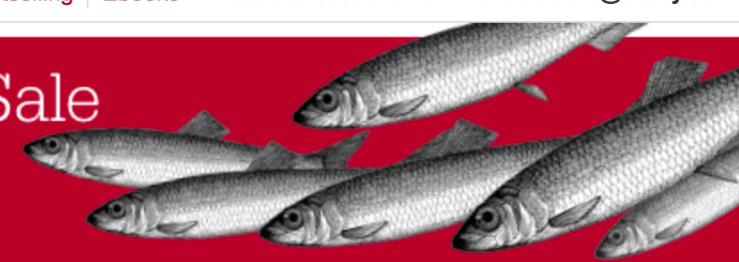
Shop Books & Videos

Search

Your Account Shopping Cart 0 items \$0.00

Home Shop Video Training & Books Radar Safari Books Online Conferences

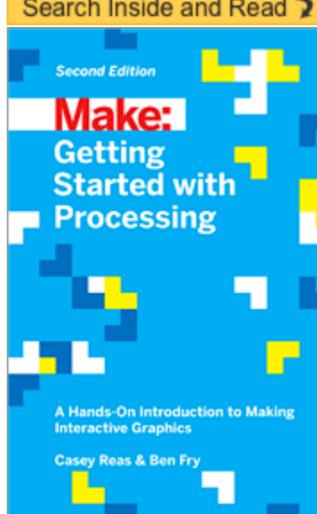
Browse Subjects Learning Paths Video Training New Upcoming Early Release Bestselling Ebooks 1-800-889-8969 / 707-827-7019 / orders@oreilly.com



Shop Our Back to (Tech) School Sale

Save 50% on all ebooks & training videos

Use discount code **B2S6** - Deal expires September 14 at 5:00am PT

Search Inside and Read 

Getting Started with Processing, 2nd Edition
A Hands-On Introduction to Making Interactive Graphics
By Casey Reas, Ben Fry
Publisher: Maker Media, Inc
Final Release Date: September 2015
Pages: 238

 5.0
[Read 1 Review](#) | [Write a Review](#)

Processing opened up the world of programming to artists, designers, educators, and beginners. This short book gently introduces the core concepts of computer programming and working with Processing. Written by the co-founders of the Processing project, Reas and Fry, Getting Started with Processing shows you how easy it is to...

[Full description](#)

 Buy 2 Get 1 FREE
 Free Shipping On Orders > \$29.95
 100% Guarantee More Info

Buying Options

Immediate Access - Go Digital what's this?

Ebook: \$12.99 [Add to Cart](#)

Formats: DAISY, ePub, Mobi, PDF

Print & Ebook: \$27.49 [Add to Cart](#)

Print: \$24.99 [Add to Cart](#)

[Safari Books Online - Read now >](#)

Essential Links

Getting Started with Proce... Contact Us: Step 1 of 2 | T... LP Learning Processing 2nd E... +

https://mitpress.mit.edu/txbkreq/243136 Search My Account Cart (0) Help

The MIT Press Books Journals Blog About Giving Search this site... GO

[COPY REQUEST](#)

To Order

Books: Call toll-free (800) 405-1619 in the US and Canada. Call (401) 658-4226 from South America and Asia.

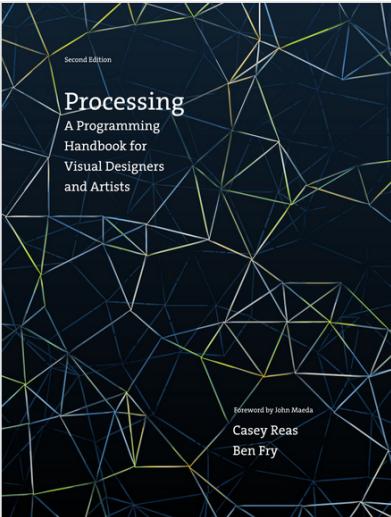
Journals: Call (617) 253-2889 for worldwide ordering. To see complete contact info, Use the contact form at right.

General Press Inquiries

The MIT Press
One Rogers Street
Cambridge, MA 02142-1209 USA
Tel: (617) 253-5646
Fax: (617) 258-6779

Textbook Request: Step 1 of 2

You are requesting:



Processing, Second Edition
by [Casey Reas](#) and [Ben Fry](#)

Please tell us the region from which you are requesting this textbook and/or instructor materials.

United States

