Learning Processing

[Website]

Chapter 1: Pixels

Color range

```
colorMode(RGB,100,500,10,255);
```

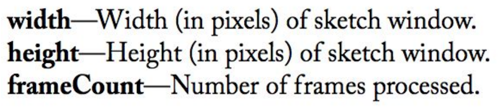
Chapter 2: Processing

Chapter 3: Interaction

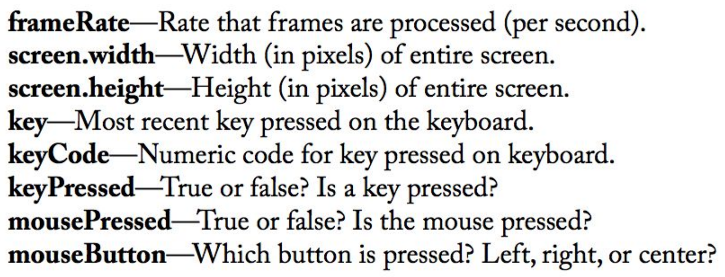
mouseX, mouseY, pmouseX, pmouseY

Chapter 4: Variables

System variables



width—Width (in pixels) of sketch window.
height—Height (in pixels) of sketch window.
frameCount—Number of frames processed.

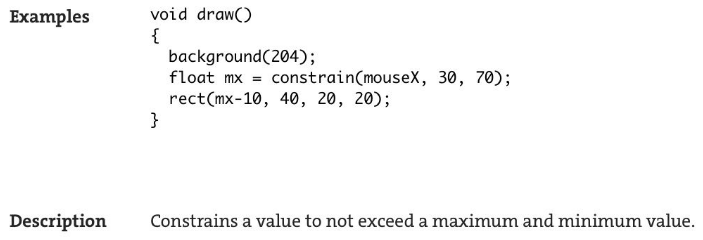


frameRate—Rate that frames are processed (per second).
screen.width—Width (in pixels) of entire screen.
screen.height—Height (in pixels) of entire screen.
key—Most recent key pressed on the keyboard.
keyCode—Numeric code for key pressed on keyboard.
keyPressed—True or false? Is a key pressed?
mousePressed—True or false? Is the mouse pressed?
mouseButton—Which button is pressed? Left, right, or center?

Chapter 5: Conditionals

Chapter 6: Loops

Use constrain() to exit loops



Examples

```
void draw()
{
  background(204);
  float mx = constrain(mouseX, 30, 70);
  rect(mx-10, 40, 20, 20);
}
```

Description Constrains a value to not exceed a maximum and minimum value.

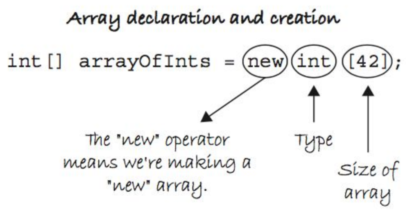
Chapter 7: Functions

Null

Chapter 8: Objects

Null

Chapter 9: Arrays


Array declaration and creation
int [] arrayOfInts = new int [42];
The "new" operator means we're making a "new" array.
Type
Size of array

Resize using [append()]

Processing frame functions: [frameRate()], [frameCount()], and [frameRate]

Chapter 10: Algorithms

[dist()]

[Max size of arrays] = 2*31 = 2147483647

[Rain drop]

Chapter 11: Debugging

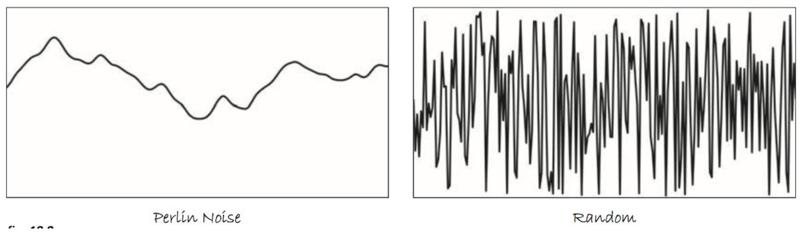
Chapter 12: Libraries

Chapter 13: Mathematics

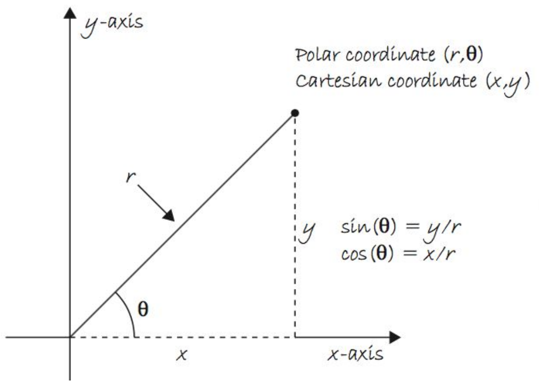
Noise generation

[Perlin Noise]

noise() vs. random



map()
How to draw circle



How to draw wave
Stroke vs. fill

Chapter 14: Transformations and 3D

translate()

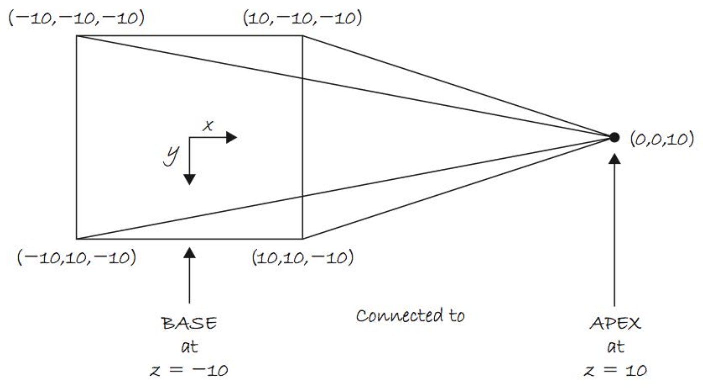
P3D vs. OPENGL

P3D—P3D is a 3D renderer developed by the creators of *Processing*. It should also be noted that anti-aliasing (enabled with the ***smooth()*** function) is not available with P3D.
OPENGL—OPENGL is a 3D renderer that employs hardware acceleration. If you have an OpenGL compatible graphics card installed on your computer (which is pretty much every computer), you can use this mode. Although at the time of the writing of this book, there are still a few, minor kinks to be worked out with this mode (you may find things look slightly different between P3D and OPENGL), it may prove exceptionally useful in terms of speed. If you are planning to display large numbers of shapes onscreen in a high-resolution window, this mode will likely have the best performance.

Custom shapes

beginShape()
vertex()
endShape()

3D Coordinates



Rotation

rotateX(), rotateY(), rotateZ()
Set origin to center

```
// Translate origin to center
translate(width/2, height/2);
```

rectMode()

The default mode is rectMode(CORNER), which interprets the first two parameters of rect() as the upper-left corner of the shape, while the third and fourth parameters are its width and height.

Example: solar system

```
// pushMatrix()
// rotate()
// translate origin
// popMatrix()
```

PShape

```
// PShape shape
// shape = createShape()
// shape.beginShape()
// shape.vertex(x, y)
// shape.endShape()
```

Chapter 15: Images and Pixels

Images

PImage

image(): The img parameter specifies the image to display and by default the a and b parameters define the location of its upper-left corner.

imageMode()

createImage(): Creates a new PImage (the datatype for storing images). This provides a fresh buffer of pixels to play with. Set the size of the buffer with the width and height parameters

random(a, b): starting at a , and up to, but not including b

Pixels

loadPixels(): Loads a snapshot of the current display window into the ***pixels[]*** array. This function must always be called before reading from or writing to pixels[].

updatePixels(): Updates the display window with the data in the ***pixels[]*** array. Use in conjunction with loadPixels().

2-D Pixel Array (use x + y * width)

```
// Loop through every pixel column
for (int x = 0; x < width; x++ ) {
  // Loop through every pixel row
  for (int y = 0; y < height; y++ ) {

    // Use the formula to find the 1D location
    int loc = x + y * width;

    // If even column
    if (x % 2 == 0) {
      pixels[loc] = color(255);
      // If odd column
    } else {
      pixels[loc] = color(0);
```

Image Processing

red(), blue(), green(), hue(), saturation(), brightness(), alpha()

filter(): Filters the display window using a preset filter or with a custom shader. Using a shader with filter() is much faster than without. Shaders require the P2D or P3D renderer in size().

tint()

Pixel Group Processing

Convolution

Convolution matrix

```
Sharpen:
 -1   -1   -1
 -1    9   -1
 -1   -1   -1

Blur:
1/9  1/9  1/9
1/9  1/9  1/9
1/9  1/9  1/9
```

Convolution example

Pointillism

Example

Explode 3D

Example

Chapter 16: Video

Processing video capture setup

1. Import the *Processing* video library

   `import processing.video.*;`

2. Declare a Capture object

   `Capture video;`

3. Initialize the Capture object

   `video = new Capture();`

4. Capture setup

```
void setup() {
  video = new Capture(this,320,240,30);
}
```

5. Read image from the camera input

```
void draw() {
  if (video.available()) {
    video.read();
  }
}
```

or

captureEvent() is a function and therefore needs to live in its own block, outside of ***setup()*** and ***draw()***.

```
void captureEvent(Capture video) {
  video.read();
}
```

6. Display the image to canvas

`image(video, x, y);`

Processing video display setup

1. Import the *Processing* video library

   `import processing.video.*;`
2. Declare a Movie object

   `Movie movie;`

3. Initialize a Movie object

   `movie = new Movie(this, “yourmovie.mov”);`
4. Start movie playing

   `movie.loop();`
5. Read frames from the movie

```
void draw() {
  if (movie.available()) {
    movie.read();
  }
}
```

Or:

```
void movieEvent(Movie movie) {
  movie.read();
}
```

6. Display the movie

   `image(movie, x, y);`

Video manipulation

// The jump() function allows you to jump immediately to a point of time within the video.
// duration() returns the total length of the movie in seconds.

`movie.jump(ratio * movie.duration());`

Video pixelation

Example

Computer vision

Color tracking

Background removal
Motion detection: example 1, example 2

Chapter 17: Text

String

Class methods

charAt()	Returns the character at the specified index
equals()	Compares a string to a specified object
indexOf()	Returns the index value of the first occurrence of a substring within the input string
length()	Returns the number of characters in the input string
substring()	Returns a new string that is part of the input string
toLowerCase()	Converts all the characters to lower case
toUpperCase()	Converts all the characters to upper case

String functions

join()
match()
matchAll()
nf()
nfc()
nfp()
nfs()
split()
splitTokens()
trim()

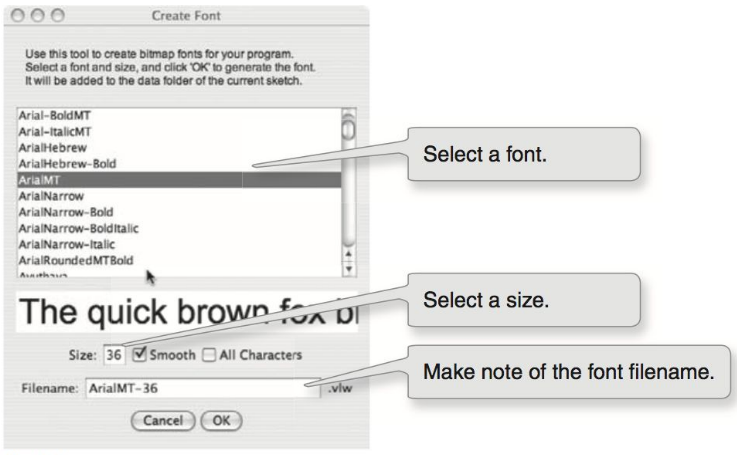
splitToken(): for multiple delimiters
join(): concatenate strings

```
String s = "a, b c ,,d ";
String[] q = splitTokens(s, ", ");
println(q.length + " values found");  // Prints "4 values found"
println(q[0]);  // Prints "a"
println(q[1]);  // Prints "b"
println(q[2]);  // Prints "c"
println(q[3]);  // Prints "d"
```

Display text

1. Create font with Processing

Choose a font by selecting "Tools" → "Create Font." This will create and place the font file in your data directory. Make note of the font filename for Step 3. *Processing* uses a special font format, "vlw," that uses images to display each letter. Because of this, you should create the font at the size you intend to display. See Figure 17.1.



Create Font

Use this tool to create bitmap fonts for your program.
Select a font and size, and click 'OK' to generate the font.
It will be added to the data folder of the current sketch.

Arial-BoldMT
Arial-ItalicMT
ArialHebrew
ArialHebrew-Bold
ArialMT
ArialNarrow
ArialNarrow-Bold
ArialNarrow-BoldItalic
ArialNarrow-Italic
ArialRoundedMTBold

The quick brown fox bi

Size: 36 ☑ Smooth ☐ All Characters

Filename: ArialMT-36 .vlw

fig. 17.1

2. Declare a PFont object

    `PFont f;`

3. Load font

    //*Processing* uses a special font format, "vlw," that uses images to display each letter. Because of this, you should create the font at the size you intend to display

    `F = loadFont("yourfont.vlw");`

4. Specify the font using textFont()

    `textFont(f, fontsize);`

5. Specify color

    `fill(color);`

6. Call text() function to display text

    `text("yourtext", x, y);`

Text formatting (attributes)

textAlign()
textWidth()
textLeading()
textShape()
textSize()

Text animations

Scrolling text
Text Mosaic
Letter shaking (breaking up)
Shape / text along a path

Chapter 18: Data Input

Table

```
Table table;

void setup() {

  table = new Table();

  table.addColumn("id");
  table.addColumn("species");
  table.addColumn("name");

  TableRow newRow = table.addRow();
  newRow.setInt("id", table.getRowCount() - 1);
  newRow.setString("species", "Panthera leo");
  newRow.setString("name", "Lion");

  saveTable(table, "data/new.csv");
}

// Sketch saves the following to a file called "new.csv":
// id,species,name
// 0,Panthera leo,Lion
```

Table Methods

addColumn()	Adds a new column to a table
removeColumn()	Removes a column from a table
getColumnCount()	Gets the number of columns in a table
getRowCount()	Gets the number of rows in a table
clearRows()	Removes all rows from a table
addRow()	Adds a row to a table
removeRow()	Removes a row from a table
getRow()	Gets a row from a table
rows()	Gets multiple rows from a table

getInt()	Get an integer value from the specified row and column
setInt()	Store an integer value in the specified row and column
getFloat()	Get a float value from the specified row and column
setFloat()	Store a float value in the specified row and column
getString()	Get an String value from the specified row and column
setString()	Store a String value in the specified row and column
getStringColumn()	Gets all values in the specified column
findRow()	Finds a row that contains the given value
findRows()	Finds multiple rows that contain the given value
matchRow()	Finds a row that matches the given expression
matchRows()	Finds multiple rows that match the given expression
removeTokens()	Removes characters from the table
trim()	Trims whitespace from values
sort()	Orders a table based on the values in a column

Table example
loadTable()
TableRow

Text Parsing

loadString(): can load both text files and HTML/XML, (blocking function)

*use loop()/noloop() to control the draw loop
*use redraw() execute the draw loop once

XML

Reference
loadXML()
parseXML()
saveXML()

XML methods

getParent()	Gets a copy of the element's parent
getName()	Gets the element's full name
setName()	Sets the element's name
hasChildren()	Checks whether or not an element has any children
listChildren()	Returns the names of all children as an array
getChildren()	Returns an array containing all child elements
getChild()	Returns the child element with the specified index value or path
addChild()	Appends a new child to the element
removeChild()	Removes the specified child
getAttributeCount()	Counts the specified element's number of attributes
listAttributes()	Returns a list of names of all attributes as an array
hasAttribute()	Checks whether or not an element has the specified attribute
getString()	Gets the content of an attribute as a String
setString()	Sets the content of an attribute as a String
getInt()	Gets the content of an attribute as an int
setInt()	Sets the content of an attribute as an int
getFloat()	Gets the content of an attribute as a float
setFloat()	Sets the content of an attribute as a float
getContent()	Gets the content of an element
getIntContent()	Gets the content of an element as an int
getFloatContent()	Gets the content of an element as a float
setContent()	Sets the content of an element
format()	Formats XML data as a String
toString()	Gets XML data as a String using default formatting

Chapter 19: Data Stream

Network

Processing network library

Network events

serverEvent(): when a new client connects to a server

- clientEvent(): called when a server sends a byte to an existing Client object
- disconnectEvent(): called when a client disconnects

Server

- Reference
  - available(): Returns the next client in line with a new message
  - write(): Writes a value to all the connected clients
  - trim(): Removes whitespace characters from the beginning and end of a String + linebreak
- Simple server example

Client

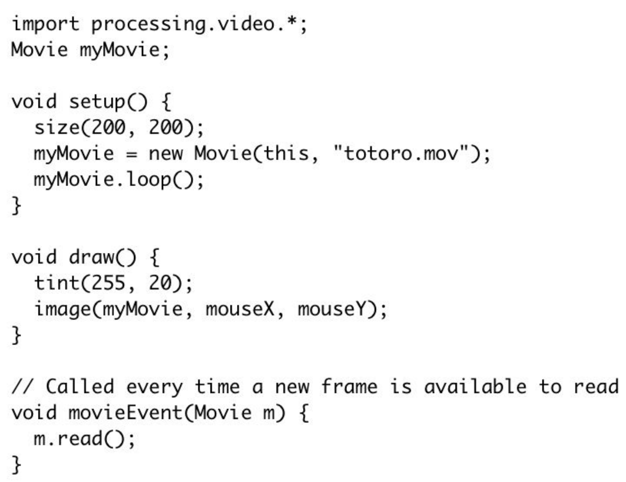
- Reference
  - available(): Returns the next client in line with a new message
  - read(): Returns a number between 0 and 255 for the next byte that's waiting in the buffer
  - readString(): Returns the all the data from the buffer as a String
  - write(): Writes a value to all the connected clients
  - trim(): Removes whitespace characters from the beginning and end of a String + linebreak
- Simple client example

Serial

- Reference
  - list(): a list of all available serial ports
  - readStringUntil()
- Simple serial example
- Serial handshaking

Chapter 20: Sound

Play audio file



```
import processing.video.*;
Movie myMovie;

void setup() {
  size(200, 200);
  myMovie = new Movie(this, "totoro.mov");
  myMovie.loop();
}

void draw() {
  tint(255, 20);
  image(myMovie, mouseX, mouseY);
}

// Called every time a new frame is available to read
void movieEvent(Movie m) {
  m.read();
}
```

Sonia
Minim

Chapter 21: Exporting

Pdf

- Reference
    - exit(): stops the pdf from rendering
- Basic example
- Live recording pdf example
    - beginRecord():Opens a new file and all subsequent drawing functions are echoed to this file as well as the display window
    - endRecord(): Stops the recording process started by beginRecord() and closes the file
- Render 3D files using openGL
- Image/saveFrame()

Chapter 22: Advanced OOP

OOP concepts

- Encapsulation
- Inheritance
    - extends: keyword to inherit a class
- Polymorphism
- Overloading

Chapter 23: Java

Userful Java classes

- ArrayList
- HashMap