Index

Symbols
+ (addition) operator, 40
/ (division sign), 50
% (modulo) operator, 554
* (multiplication sign), 50
=== (strict equality) operator, 5
!== (strict inequality) operator, 5

A
acceleration, xxxi, 36, 59–69
    accumulation of forces, 77–78
    angular, 120–121, 123–124, 155, 157–158
    fluid resistance, 100
    gravitational attraction, 87
    Newton's second law of motion, 75–77
    vector, 59–69
accept-reject algorithm, 18–19, 455
activation functions, 504–505, 508, 520
Adaptation in Natural and Artificial Systems (Holland), 438
adaptive decision-making systems, 499
addForce() method, 355, 357
addition (+) operator, 40
additive blending, 209–210
add() method, 42, 45, 65, 298–299
addParticle() method, 338
air resistance, 94. *See also* fluid resistance
alignment in flocking, 259, 268–269, 271
align() method, 270
alive property, 559–560
amplitude
    defined, 135
    oscillation, 135–137
        with angular velocity, 140–142
    waves, 143
anchors
    distance constraints, 316–317
    revolute constraints, 319
    spring forces, 147–150
Anderson, Carl D., 167
Anderson, Edgar, 524–525
angleBetween() method
    dot products, 242–243
    path following, 248
    vectors, 45
angleMode() function, 119, 281
angles, 118–120
    angular acceleration, 120–121, 123–124, 155, 157–158
    angular motion (rotation), 119–130
    angular velocity, 120–122, 124, 138–142, 155, 157, 161
    degrees, 118–119
    radians, 119
angle variable, 122, 139–140, 144, 160
Animal class and objects, 188–192
anomaly detection, 499
applyBehaviors() method, 266
applyForce() method
    creating forces, 83, 85
    factoring in mass, 81
    fluid resistance, 95
    force accumulation, 78
    genetic algorithms, 471
    gravitational attraction, 101, 103, 105
    Newton's second law, 77
    particle systems, 170, 172, 197–198
    physics engines, 324–325
    spring forces, 149, 151
    steering forces, 220
applyRepeller() method, 202
arccosine, 128, 242
arcsine, 128
arctangent, 128–129
Aristotle, 73
Array class and objects, 174–175
array destructuring, 415
arrive behavior, 224–229
arrow notation, 40, 178
artificial intelligence (AI), 501. *See also* machine learning; neural networks
ASCII (American Standard Code for Information Interchange), 451
Ashley, Quinton, 291
associative rule, 42, 50
asynchronous operations, 534
async keyword, 534
atan() function, 128–129
atan2() function, 129–130
AttractionBehavior class and objects, 355–357
attract() method, 106, 111, 326
Attractor class and objects, 103–111, 151–152, 325, 356
autonomous agents, xxxii, 213–285
    algorithmic efficiency, 274–284
    complex systems, 215, 257–274
    defined, 214
    flow-field following, 233–239
    key components of, 214–215, 220
    path following, 240–257
    steering behaviors, 215–240, 260–265, 268–274
await keyword, 534

B
background() method, 368
backpropagation, 520
Barnes-Hut algorithm, 115, 279
beginShape() function
    polygons, 311
    soft-body simulations, 344
    vehicles and steering, 223
bell curves, 13–14

bias input, 507–508, 519–520, 526
big *O* notation, 274–275
bin-lattice spatial subdivision (binning), 275–278, 357
`Bird` class and objects, 550, 552, 555–556, 559, 563
`blendMode()` function, 209
blend modes, 208–209
`Bloop` class and objects, 488–489, 491–492
`Bob` class and objects, 150, 153
`Body`/`Bodies` class and objects, 294, 297–299
  collision events, 330
  linking `Box` objects with, 305
  n-body problem, 110–111
  polygons, 309
  static, 307
`Boid` class and objects, 268–269, 272, 275, 277
boids model, 216, 268–278, 500
`bounceEdges()` method, 93–94
`boundaries()` method, 232
`Boundary` class and objects, 307–308, 331
Box2D, 290–291, 332
`Box` class and objects, 302–303, 305–307, 323–325
`box` variable, 298
`brain` property, 555
Braitenberg, Valentino, 216
`branch()` function, 422–424, 426
Brummitt, Liam, 291
brute-force algorithms, 439–441
Burges, Christopher J.C., 523
Bushnell, Nolan, 550
butterfly effect (nonlinearity), 258

C
`calculateDrag()` method, 98
`calculateState()` method, 392
Calder, Alexander, 71
callback functions, 328–330, 534, 538, 556–557
Cantor, Georg, 401–402
`cantor()` function, 408–409
Cantor set, 402, 407–410, 431
`canvas` variable, 300
Cartesian coordinates, 130–134
  converting to polar, 131–133
  defined, 130
  evolutionary computing, 473
  path following, 252
  pendulums, 160
Carver, George Washington, 543
`Cat` class and objects, 188–190
categories. *See* labels
Catto, Erin, 290
CDNs (content delivery networks), 291–292
`Cell` class and objects, 390
cellular automata (CA), xxxii, 359–396
  cells, 360, 390–392
  classifications, 379–381
  defined, 360
  elementary (1D CAs), 362–379
  Game of Life (2D CAs), 381–390
  grids, 360–362
  neighborhoods, 360–364
  states, 360–363
  variations on, 392–395
central processing units (CPUs), 559
Chakri Maha Prasat Hall, 397
`checkTarget()` method, 480
`choice` variable, 5
`circle()` function, 44, 295, 337
classes, defined, 3. *See also* labels
classification, Wolfram, 379–381
  complexity, 380–381
  randomness, 380
  repetition, 380
  uniformity, 379
classification in machine learning, 522–529, 547
`classifier` variable, 532–533, 537
`classify()` method, 537–539, 556–557
`classifySync()` method, 557
cloning, 447, 491, 580
`Cluster` class and objects, 351–353
Coding Train, xxix–xxx, xxxviii
coefficient of friction (*Q*), 91, 94, 96
cohesion in flocking, 259, 268–269, 271–272
`collides()` method, 552
collisions
  attraction and repulsion behaviors, 356
  callbacks, 328–330, 534
  collision detection, 289–290, 329
  collision events, 327–331
  collision resolution, 289
  event listeners, 328
  idealized elastic collisions, 92
  inelastic collisions, 92–93
  neuroevolution, 552–553, 564
  physics libraries, 334
`colorMode()` method, 484
comma-separated values (CSV), 531
commutative rule, 42
complex adaptive systems, 395, 500
complex systems, 215, 257–274
  Class 4 cellular automata, 380–381
  combining behaviors, 265–268
  competition and cooperation, 258–259
  defined, 258
  feedback, 259
  flocking, 268–274
  Game of Life, 382
  implementing group behaviors, 259–265
  key principles and characteristics, 258
  nesting, 395
  nonlinearity, 258
  separation, 260–265
`Composite` container, 294, 298
compound bodies, 309, 312–314
Computational Beauty of Nature, The (Flake), xxx
concave shapes, 310
`Confetti` subclass and objects, 194–195, 197
connectionist systems, 500
constant acceleration, 60–62
`const` declaration, 5, 297
`constrain()` function
  angular motion, 125
  steering behaviors, 237
`Constraint` class and objects, 315, 317, 319
constraints, 294, 315–324
  distance constraints, 315–319

mouse constraints, 315, 322–324
revolute constraints, 315, 319–322
`constructor()` method
particle systems, 197
random walkers, 4
constructors, defined, 4
`contactEdge()` function, 92
`contains()` method, 98, 479–480
content delivery networks (CDNs), 291–292
convex shapes, 310
Conway, John, 381
`copy()` method
forces and mass, 81
neuroevolutionary ecosystem simulation, 580
reproduction, 492
Cortes, Corinna, 523
`cos()` function
coordinate conversion, 131
path following, 248
pendulums, 160
cosine
coordinate conversion, 131, 160
defined, 126
dot products, 242–243
lookup tables, 280–282
oscillation, 137
path following, 247
pendulums, 156
Coulton, Jonathan, 397
Courville, Raphaël de, xxx
Coveyou, Robert R., 1
CPUs (central processing units), 559
`createButton()` method, 571
`createCanvas()` method, 210
`create()` function, 296, 299, 312, 323
`createVector()` function, 39–40, 140, 297
`Creature` class and objects, 568, 570, 572, 574, 580
creature design, 585–588
matching form to environment, 587
perspective, 586
varying elements, 586–587
visual elements, 585
`cross()` method, 45
crossover, 447–450, 457–459, 561–563
`crossover()` method, 458, 492, 562
custom distributions, 16–19
accept-reject algorithm, 18–19
avoiding oversampling, 17
Lévy flight, 17
qualifying random values, 17–18
`CustomShape` class and objects, 309, 311

D
`debug` property, 532, 534
degrees, 118–119
delta angle, 143
`deltaAngle` variable, 143–145
delta time, 79, 121
delta weight, 510
dense (fully connected) layer, 526
derivatives, 331
deterministic structure, 24, 399–400, 402, 419–424, 426
differentiation, 331
dissipative forces, 90
distance constraints, 316–319
`dist()` method, 45
distributive rule, 50
division sign (/), 50
`div()` method, 45, 50, 65
`DNA` class and objects, 451–454, 458, 460–461, 467–469, 472–475, 484, 491–492
`Dog` class and objects, 187–192
`dot()` method, 45, 241
drag force, 94–95. *See also* fluid resistance
`drag()` method, 98
`drawCircles()` function, 405–406

E
`eat()` method, 187, 192–193, 580
Ecosystem Project, xxxvii–xxxviii
autonomous agents, 285
cellular automata, 396
forces, 116
fractals, 435
genetic algorithms, 495
neural networks, 542
neuroevolution, 582
oscillation, 165
particle systems, 211
physics engines, 358
randomness, 31
vectors, 69
ecosystem simulation, 439, 487–493
genotype/phenotype distinction, 444, 490
neuroevolution, 573–582
reproduction, 491–493
selection, 491–493
elastic collisions, 92
elitist approach to parent selection, 445
`elt` property, 300
Emerson, Ralph Waldo, 213
`Emitter` class and objects, 179–184, 186–187, 195–198
emitters, 170, 179–181, 182–185
`emitters` variable, 184–185
`endShape()` function, 223, 311, 344
`Engine` class and objects, 294, 296–297, 304, 326
epochs, 533–534, 536
equilibrium, 73
Euclid, 34, 398
Euclidean geometry, 398, 401
Euclidean vectors, 34. *See also* vectors
Euler, Leonhard, 332
Euler integration, 332
event listeners, 328
evolutionary computing, 438. *See also* genetic algorithms
exclusive or (XOR), 518–520
exponential increase, 465
`extends` keyword, 190

F
factorial() function, 404–405
factorials, 403–404
factory methods, 297–298
feedForward() method, 508
feed-forward model, 503
filter() function, 178
finishedTraining() callback function, 534, 538
fitness functions
    customizing GAs, 464–467
    defined, 445
    ecosystem simulation, 489, 579
    interactive selection, 483
    reproduction, 448
    selection, 452–453
    smart rockets, 471–472, 481, 570
Flake, Gary William, xxx
flap() method, 550, 553, 557
Flappy Bird game, 545–565
    classification, 547
    coding, 550–554
    collision events, 552, 564
    decision-making process, 555
    features, 546–547, 555–556
    heredity, 561–565
    neural network, 546–548
    neuroevolution, 554–565
    Nolan's law, 550
    normalizing scores, 561, 563
    reinforcement learning, 545
    reproduction, 561–565
    selection, 559–561
    variation, 557
fleeing behavior, 223, 261–265
Flock class and objects, 272
flocking, 268–274
    rules governing, 268–269
    short-range relationships, 270
flock() method, 269
floor() method, 5, 7
Flower class and objects, 484
FlowField class and objects, 237–239
flow-field following, 233–239
fluid resistance, 94–100
    formula for, 94–96
    Mover and Liquid objects, 96–100
    objects of variable mass, 98–100
    simplified formula for, 96
Food class and objects, 492, 575
forces, xxxi, 71–116
    accumulation of, 77–79
    action/reaction pairs, 74
    creating, 82–88
    defined, 72
    equilibrium, 73
    mass and, 79–82
    modeling, 82, 88–110
    n-body problem, 110–115
    Newton's laws of motion, 72–77
    particle systems with, 197–200
    physics engines, 72, 324
    time step, 79
force variable, 149
forEach() loops, 175
for...in loops, 175
for loops
    arrays of particles, 174, 176
    cellular automata, 370–371
    defined, 175
    L-systems, 429
    neuroevolutionary steering behaviors, 571
    recursive functions, 403, 405–406, 408
    soft-body simulations, 343
    waves, 142, 144
for...of loops
    arrays of particles, 175–176
    collision events, 329
    defined, 175
    Koch curve, 413
    particle emitters, 185
    particle systems with forces, 199
    polygons and shapes, 311
Fractal Geometry of Nature, *The* (Mandelbrot), 398, 401
fractals, 397–435
    Cantor set, 402, 407–410, 431
    defined, 398
    deterministic structure, 399–400
    Euclidean geometry vs., 398, 401
    examples of, 398–400
    Koch curve, 411–419
    L-systems, 427–435
    Mandelbrot set, 401
    mathematical monsters, 412
    production rules, 401–402
    recursion, 401–410
    self-similarity, 400
    Sierpiński triangles, 366, 373
    stochastic structure, 400
    tree structure, 399, 419–427
frameCount variable, 136–137, 179, 554
frames
    arrays of particles, 179
    number elapsed, 136
    oscillation, 136–138, 140
    pendulums, 157, 161
    waves, 145
frequency, defined, 138
friction, 90–94
    calculating, 92
    coefficient of, 91, 94, 96
    dissipative force, 90
    elastic vs. inelastic collisions, 92–93
    formula for, 89–91
    velocity vector, 90–91
fromAngle() method, 133
fromCharCode() method, 451
fromVertices() function, 309
fully connected (dense) layer, 526
function keyword, 4

G
G (universal gravitational constant), 101–103, 107
Gala (Riley), 117
Galapagos (Sims), 483–484
Galileo, 87
Game of Life, 381–392
    implementation, 385–390

object-oriented cells, 390–392
rules of, 382–385
Gardner, Martin, 381
GAs. *See* genetic algorithms
Gauss, Carl Friedrich, 13
Gaussian distributions. *See* normal distributions
geneCounter variable, 474
generate() function, 414, 417–418, 429
generations. *See also* reproduction; selection
cellular automata, 362–363
defined, 362
L-systems, 428
speeding up, 570
generation variable, 377
generative models, 500
genetic algorithms (GAs), xxxii, 437–495
coding, 450–462
customizing, 463–469
defined, 438–439
ecosystem simulation, 439, 487–493
evolution vs. intelligent design, 438
genotype/phenotype, 443–444, 467–469, 472, 490
heredity, 442, 447, 561–565
interactive selection, 439, 482–487
population creation, 443–444, 450–452
reproduction, 447–450, 457–459, 463–464, 491–493, 561–565
selection, 442, 444–447, 450, 452–457, 491–493, 559–561
smart rockets, 469–482
utility of, 440–441
variation, 442–443, 445–447, 449, 557–559
genotype
customizing GAs, 467–469
defined, 443
ecosystem simulation, 490
genotype/phenotype distinction, 444
interactive selection, 483, 486
smart rockets, 472
geometric vectors, 34. *See also* vectors
gesture classifiers, 529–541
data collection, 530–531
data preparation, 530–531
model deployment, 537–539
model evaluation, 534–537
model selection, 531–532
model training, 532–534
parameter tuning, 537
getNormalPoint() function, 252
GitHub
feedback and corrections, xxxviii
Magic Book project, xxix
repository for book, xxviii–xxix, xxxv, xxxviii, 583
global variables
avoiding unnecessary p5.Vector objects, 283
genetic algorithms, 463–464, 471, 474
gesture classifiers, 538
oscillation, 139
random walks, 6
Glow class and objects, 568, 570
graphics processing units (GPUs), 559
gravity and gravitational attraction, 101–110
acting on two objects, 86
constraining distance, 107
direction of attraction force, 102
formula for, 101
inverse proportion to distance, 101–102
n-body problem, 110–115
one object attracts another object, 103–107
one object attracts many objects, 109–110
particle systems with forces, 197–199
pendulums, 154–159
repellers vs., 200–201
scaled by mass, 88
weight, 76
GravityBehavior class and objects, 336
gravity variable, 296
G variable, 102–103, 107

H

HALF_PI constant, 119
handleCollisions() function, 328–330
handPose() function, 528
heading() method, 45, 130, 223, 530
health variable, 489, 579
heredity, 442, 447, 561–565
higher-order functions, 175, 178
hinges, 319
Hodgin, Robert, 209
Holland, John, 438
Hooke, Robert, 147
Hooke's law, 147–148
Hugo, Victor, 287
hyperparameters, 522, 534, 537

I

I (moment of inertia), 124, 158
if statements, 231, 249, 387, 530
imageClassifier() function, 528
image textures, 205–211
additive blending, 209–210
blend modes, 208–209
Gaussian distributions, 207–208
PNG format, 206
renderers, 210
resolution, 208
img variable, 207
inelastic collisions, 92–93
infinite monkey theorem, 440
Infinity value, 256–257
inheritance, 185–192
defined, 187
extending, 190–191
overriding, 191
particles with, 193–197
springs, 339
subclasses, 188, 190
superclasses, 188, 190
tree structure, 189
instanceof operator, 331
integrals, defined, 331

integration, 331–333
  defined, 331
  Euler integration, 332
  Runge-Kutta integration, 333
  Verlet integration, 333
interactive motion, 66–69
interactive selection, 482–487
  defined, 439, 483
  population creation, 483–484
  user-assigned fitness ratings, 484, 486
inverse proportion, 75, 101–102
iris dataset, 524–525
`isDead()` method, 179, 197
`isStatic` property, 300, 307

J
Jakobsen, Thomas, 333
JavaScript, xxvii–xxviii. *See also names of specific libraries and programming elements*
  abundance of options, 174
  addition operator, 40
  arrays, 174, 187, 233
  arrow notation, 178
  callback functions, 534
  `const` vs. `let`, 5
  `Infinity` value, 257
  inheritance, 190
  object destructuring, 297
  object literals, 298, 434
  objects, 3
  strict equality and inequality operators, 5
JSON (JavaScript Object Notation), 531

K
k (spring constant), 148
Kaku, Michio, 497
kente cloth, 359
khipu devices, 497
Kleinman, Kim, 525
Klise, Steven, xxix, 385
Koch, Helge von, 411
Koch curve, 411–419
`KochLine` class and objects, 412–415
`kochPoints()` method, 415
Kutta, Martin, 333

L
labels, 522–524, 526, 538, 540
Lasky, David, 585
learning constant, 511
Learning Processing (Shiffman), 185
LeCun, Yann, 523
`lerp()` method, 45
`let` declaration, 5
Lévy flight, 17
`lifeCounter` variable, 477–478
`lifespan` property, 170–171, 392, 474, 477
`limit()` method, 45, 61
Lindenmayer, Aristid, 427
Lindenmayer systems. *See* L-systems
linearly separable problems, 517–519
`line()` function, 408–409, 412, 421–422, 432
`Liquid` class and objects, 96–99
`liquid` variable, 97
living root bridges, 287
`load()` function, 537
`loadJSON()` function, 534
`lock()` method, 340
`lookup()` method, 237–239
lookup tables, 280–282
Lorenz, Edward Norton, 258
L-systems (Lindenmayer systems), 427–435
  alphabet, 428
  axiom, 428
  generations, 428
  production rules, 428
  simple, 428–430
  strings, 427–428
Lunar Lander game, 549–550

M
machine learning. *See also* neural networks
  classification, 522–529
  epochs, 533–534, 536
  features, 546–547
  life cycle, 521–522, 529–541
  loss, 536
  with ml5.js, 521–529
  network design, 524–528
  regression, 524, 527–528
  reinforcement learning, 501, 545–549
  supervised learning, 501, 509, 521–523, 549
  transfer learning, 502
  unsupervised learning, 501
Madsen, Rune, xxix
`mag()` function, 45, 51, 95
Magnetosphere, 209
magnitude squared, 279–280
`magSq()` function, 280
Mandelbrot, Benoit, 398, 401
Mandelbrot set, 397, 401
`map()` function
  genetic algorithms, 484
  genotype/phenotype, 468
  neuroevolutionary ecosystem simulation, 576
  noise ranges, 21–22
  oscillation, 136
  particle systems, 195
Marsh, Zannah, 31, 585–588
Marshall Islands stick chart, 33–69
mass
  fluid resistance and objects of variable mass, 98–100
  force accumulation, 77–79
  gravitational attraction, 101–102, 104, 109
  incorporating into simulations, 79–82
  Newton's second law, 75–77
  units of measurement, 80
  weight vs., 76
`mass` variable, 79, 83, 197
mathematical monsters, 412
mating pools, 445, 453–455
Matter.js library, 288, 291–328
  adding forces, 324–327
  `Body` objects, 294, 297–299
  collision events, 327–331

compared to Toxiclibs.js, 334–335
`Composite` container, 294
constraints, 294, 315–324
`Engine` objects, 294, 296
importing, 291–293
object destructuring, 297
overview of, 293–296
p5.js and, 302–305
polygons and groups of shapes, 309–315
`Render` class, 299–301
static bodies, 307–308
vectors, 294–295
McCulloch, Warren S., 499
mean
calculating, 14–15
defined, 13
methods
defined, 4
static vs. nonstatic, 64–66
Miikkulainen, Risto, 550
Million Random Digits with 100,000 Normal Deviates, *A* (RAND), 1
`millis()` function, 136
Mills, Mike, xxvii
ml5.js library, 502
machine learning, 521–529
model deployment, 537
model selection, 531–532
model training, 532
neuroevolution, 554–557, 559, 562–563, 566–567
parameter tuning, 537
reinforcement learning, 548
splitting datasets, 537
syntax, 528–529
MNIST (Modified National Institute of Standards and Technology) dataset, 523–524
modulo (`%`) operator, 554
mo'i fish, 213
moment of inertia (*I*), 124, 158
`Mouse` class and objects, 323
`MouseConstraint` class and objects, 315, 323–324
mouse constraints, 315, 322–324
`mousePressed()` function, 328, 485
`Mover` class and objects
acceleration, 60–61
angular motion, 122–123, 126–128
combining steering behaviors, 266
creating forces, 82–83, 85–86
fluid resistance, 96–98
friction, 92
gravitational attraction, 103–110
incorporating mass into simulations, 79, 81
interactive motion, 67
motion with vectors, 55–57, 59
Newton's first law, 73
Newton's second law, 76–77
spring forces, 149
Muhonen, Taru, xxx
multilayered perceptrons, 519–520
multiplication sign (`*`), 50
`mult()` method
combining steering behaviors, 267
interactive motion, 68
vector multiplication, 45, 48–49, 65
`mutate()` method, 458–459, 562
mutation, 449–450, 457–459, 463–464, 562

N

N (normal force), 91–92
namespaces, 295
natural language processing (NLP), 500
n-body problem, 110–115
arrays, 113–115
two-body attraction, 111–113
NEAT (NeuroEvolution of Augmenting Topologies) algorithm, 550
neighborhoods, 360–364, 367, 371–374, 381–383
`neighborSum` variable, 387
`neuralNetwork()` function, 528, 534, 555, 563, 567
neural networks, xxxii, 497–542
adaptive nature of, 500
backpropagation, 520
classification, 522–529
data normalization, 517
defined, 498
difficult applications for artificial neural networks, 499–500
epochs, 533–534, 536
gesture classifiers, 529–541
human brain, 498–499
learning strategies, 501, 502
loss, 536
machine learning
defined, 501
libraries, 502, 521–529
life cycle, 521–522, 529–541
network design, 524–528
neurons, 498–499
nonlinearly separable problems, 518
perceptrons, 502–520
pretrained models, 502
procedural vs. connectionist systems, 500
regression, 524, 527–528
synthetic data, 512
weight adjustment, 500–501
neuroevolution, xxxii, 543–582
collision events, 552, 564
decision-making process, 555
defined, 544
early examples of, 549
ecosystem simulation, 573–582
features, 555–556
heredity, 561–565
NEAT algorithm, 550
normalizing scores, 561, 563
reinforcement learning and, 545–549
reproduction, 561–565
selection, 559–561
steering behaviors, 565–573
variation, 557
`neuroEvolution` property, 555
New Kind of Science, *A* (Wolfram), 361

new operator, 6
Newton, Isaac, xxxi, 72
Newton's laws of motion, 72–77
  first law, 72–73
  second law, 75–77, 80, 89, 155, 157–158
  third law, 73–75
nextGeneration() function, 485
Nguyen, Dong, 550–554
noiseDetail() function, 27
noise() function, 20–21, 27, 139
nonlinearity (butterfly effect), 258
nonlinearly separable problems, 518–519, 526
nonstatic methods, 64–66
nonuniform distributions, 9–13
  applying unequal weights to multiple outcomes, 11–12
  asking for random numbers, 11
  controlling probability of events, 13
  filling arrays with numbers, 11
  uses for, 9–10
normal distributions (Gaussian distributions), 13–16, 207
  bell curves, 13–14
  defined, 13
  standard, 15
normal force (*N*), 91–92
normalization
  data, 517, 533
  fitness scores, 445–446, 561, 563
  vectors, 53–55, 67, 102–103, 295
normalizeData() function, 533
normalizeFitness() function, 563
normalize() method
  interactive motion, 68
  path following, 248
  vector normalization, 45, 53–54
  velocity unit vector, 96
n-squared algorithms, 115
Nucera, Diana, 501

O
object destructuring, 297, 415
object literals, 298, 316–317, 434, 555
object-oriented programming (OOP), 3–4, 59, 83, 158–159, 239. *See also* inheritance; polymorphism
objects, defined, 3
Obstacle class and objects, 478–480
offscreen() method, 553
one-dimensional Perlin noise, 20–22
Onuoha, Mimi, 501
optimization, 522
oscillation, xxxi, 117–165
  angles, 118–120
  angular motion (rotation), 119, 120–130
  with angular velocity, 138–142
  defined, 118, 134
  pendulums, 154–164
  polar vs. Cartesian coordinates, 130–134
  properties of, 134–138
  spring forces, 147–154
  trigonometry functions, 125–126
  waves, 142–146
Oscillator class and objects, 140–141, 165
overfitting data, 536
oversampling, 17

P
p5.Image class and objects, 170
p5.js library, xxvii–xxviii
  const and let, 5
  documentation, xxix
  "Get Started" page, xxix
  list of sketches, xxxv
  Matter.js and, 302–305
  noise, 27
  vectors, 37–40
  Web Editor, xxxv
p5play library, 291
p5.Vector class and objects, 37, 42
  acceleration, 67–68
  angular motion, 130
  avoiding unnecessary, 282–284
  coordinate conversion, 133
  dot products, 241
  genetic algorithms, 473
  incorporating mass into force simulations, 81–82
  Koch curve, 414–415
  oscillation, 140, 142
  particle systems, 198
  steering forces and behaviors, 218, 235
  vectors, 39–40, 44, 47–48, 50–51, 53, 61, 64–66
parameter tuning, 522, 537
parseInt() function, 374
Particle class and objects, 59
  attraction and repulsion behaviors, 355–356
  collision events, 329–331
  constraints, 316, 320
  force-directed graphs, 351–352
  particle systems, 169–170, 172–175, 186, 193–199, 203
  soft-body simulations, 343, 346–347
  Toxiclibs.js particles, 337–339
ParticleString2D class and objects, 343
ParticleSystem class and objects, 59, 169, 179
particle systems, 167–211
  arrays of particles, 174–179
  defined, 168
  emitters, 179–185
  with forces, 197–200
  image textures, 205–211
  inheritance, 185–197
  polymorphism, 185–187, 192–197
  with repellers, 200–205
  single particles, 169–173
  situations involving, 168–169
Path class and objects, 245–246, 252, 254–255
path following, 240–257
  defined, 240
  dot products, 240–243
  multiple segments, 252–257
  normal, 246–247
  path radius, 245
  scalar projection, 249
  simple, 243–252

pattern recognition, 499, 505–508
Pedercini, Paolo, 291
Pendulum class and objects, 158–159
pendulums, 154–164
    angular acceleration and velocity, 155, 157
    damping trick, 161
    defined, 154
    distance constraints, 317–318
    force of the pendulum, 156–157
    gravity and tension, 154–155
    moment of inertia, 158
    net force, 155
    object-oriented structure, 159–160
    soft, 343–345
    torque, 158
People's Guide to AI, A (Onuoha and Nucera), 501
Perceptron class and objects, 508–511
perceptrons, 502–517
    activation functions, 504–505
    bias input, 507–508
    coding, 508–517
    computing output, 504–505
    defined, 502–503
    feed-forward model, 503
    learning constant, 511
    linearly vs. nonlinearly separable problems, 517–519
    multilayered, 519–520
    pattern recognition, 505–508
    perceptron algorithm, 504–505
    summing inputs, 504–505
    training, 512–515, 520
    weighting inputs, 503–505
period
    defined, 135–136
    frame count, 136–138, 179
    frequency and, 138
    oscillation, 135–138, 140
    waves, 143
Perlin, Ken, 19–20
Perlin noise, 19–30
    defined, 19
    ecosystem simulation, 488
    fractals, 427
    neuroevolutionary steering behaviors, 568
    noise ranges, 22–25
    one-dimensional, 20–22
    overusing, 29
    steering behaviors, 236, 238
    two-dimensional, 25–30
    uniform distributions vs., 19–20
    uses for, 29
phenotype
    customizing GAs, 467–469
    defined, 443–444
    ecosystem simulation, 490–491
    genotype/phenotype distinction, 444
    interactive selection, 484, 486
    smart rockets, 472, 474
Phillips, Kyle, 333
physics engines, 72, 79, 288. *See also* forces; physics libraries
physics libraries, xxxii, 287–358
    Box2D, 290–291
    integration methods, 331–333
    Matter.js library, 291–328
    p5play library, 291
    reasons for using, 289–290
    Toxiclibs.js library, 333–357
pi (π), defined, 120
PI constant, 119–120
Pipe class and objects, 551–553
Pitts, Walter, 499
pivots, 154, 158–161
plugin objects, 329–330
polar coordinates, 130–134
    converting to Cartesian, 131–133
    defined, 130
    evolutionary computing, 473
    pendulums, 160
polygon() method, 309
polymorphism, 185–187, 192–193
    defined, 187, 193
    particles with, 193–197, 338
    springs, 339
pop() function, 420–422
Population class and objects, 475–476, 485
Positron (Anderson), 167
predict() method, 566
predictSync() method, 566
preload() method, 207
pretrained machine learning models, 502, 528, 559
probabilistic approach to parent selection, 445–446
probability of the fittest, 10
procedural systems, 500
Processing, xxvii–xxix, xxxviii, 333
production rules
    fractal trees, 419–420
    Koch curve, 411
    L-systems, 427–429
    recursion, 401–402
promises, 534
pseudorandom numbers, 1, 9, 19
Pueblo pottery, 437
push() method, 175, 420–422
Pythagorean theorem, 51, 280

Q
quadratic increase, 465, 472
quadtree data structure, 278–279

R
radians, 119–120
radians() function, 119
RAND Corporation, 1
random2D() method, 45, 63–64, 235, 473
random3D() method, 45
random acceleration, 62–64
randomCharacter() function, 451, 459
random() function
    custom distributions, 17
    fractal trees, 426
    genetic algorithms, 455
    nonuniform distributions, 9, 11, 13
    normal distributions, 13
    Perlin noise, 20–21, 27
    random walkers, 4–5, 7–9

randomGaussian() function
  normal distributions, 14–16
  particle systems, 208
randomness, xxxi, 1–31
  cellular automata, 380
  custom distributions, 16–19
  nonuniform distributions, 9–13
  normal distributions, 13–16
  Perlin noise, 19–30
  pseudorandom numbers, 9
  random walks, 2–9, 63
  single-event probability, 10
  steering behaviors, 235
random walks and walkers, 2–9
  custom distributions, 16–17
  defined, 2–3
  neuroevolutionary steering behaviors, 568
  nonuniform distributions, 12–13
  Perlin noise, 23
  probability, 8
  random acceleration, 63
  uniform distributions, 8–9
  uses for, 3
raycasting, 575
recordDistance variable, 480
rectangle() method, 298, 305, 307
Rect class and objects, 336
rect() function, 298
recursion, 401–402
  Cantor set with, 407–410
  exit conditions, 404–405
  factorials, 403–404
  recursive circles, 405–407
  recursive functions, 402–404
regression, 524, 527–528, 547, 565–566
reinforcement learning, 545–549, 555, 559
  defined, 501
  features, 546–547
  policies, 549
  reward functions, 549
  supervised learning vs., 549
removeBody() method, 306–307
Render class and objects, 299–302, 305
renderers, 210
Repeller class and objects, 200–205
repellers, 200–205
repel() method, 202
reproduce() method, 580
reproduction, 447–450, 457–459, 491–493
  cloning, 447
  crossover, 447–449, 450, 457–459, 561–563
  mutation, 449–450, 457–459, 463–464, 562
  repetition, 450
reproduction() method, 475, 563
resetPipes() function, 564
Resnick, Mitchel, 215, 385
rest length, 147–148, 150, 339
revolute constraints, 319–322
Reynolds, Craig, 215–216, 218, 220–221, 227–229, 233, 238, 240, 243, 249–250, 253, 257, 261, 263, 265, 268, 275
rigid-body simulations, 334, 342. *See also* Matter.js library
Riley, Bridget, 117
Rocket class and objects, 469–472, 474–475, 479–480
rollover() method, 485
Ronald, Edmund, 549–550
Rosenblatt, Frank, 502–503
rotate() function, 45, 420–422
  angular motion, 118, 121–122
  physics engines, 306
  turtle graphics, 432
rulesets, 363–365, 367–370, 373–374, 379
  arbitrary nature of, 364
  defined, 363
  storing, 373
rules() function, 370, 372–374
Runge, Carl, 333
Runge-Kutta integration, 333
run() method
  arrays of particles, 174
  flocking, 272
  genetic algorithms, 479
  neuroevolutionary steering behaviors, 566
  particle emitters, 184
  particle systems, 197
Runner class and objects, 300–302, 304

S
Samuel, Arthur Lee, 501
Satoro, Ryunosuke, 359
save() function, 537
scalar projection, 243, 249, 252
scalars
  angular motion, 121
  formulas, 89
  mass, 80
  vector multiplication, 48
  vectors vs., 40
scaling, 48
Schmidt, Karsten, 333
Schoenauer, Marc, 549–550
<script> tags, 291–292, 334, 521
seeking behavior, 217–224, 260–261
seek() method, 225, 249–250
  combining with separate(), 266–267
  neuroevolutionary steering behaviors, 568–569
  reducing number of temporary objects, 283
  seeking behavior, 219
selection, 442, 444–447, 450, 452–457, 491–493
  elitist approach, 445
  fitness functions, 445
  interactive, 439, 482–487
  mating pools, 445, 453–455
  normalizing scores, 445–446
  parent selection, 445–446, 454–455
  probabilistic approach, 445–446
selection() method, 475, 477–478
Sensor class and objects, 576
sensors
  creating, 573–578

learning from, 578–581
soft, 499
separate() method
combining with seek(), 266–267
flocking, 271
separation, 260–262, 264
separation in flocking, 259–265, 268–269
setBackend() function, 558–559
setMag() method, 45, 68, 103
shapes
compound bodies, 312–314
concave shapes, 310
convex shapes, 310
creature design, 585
of data, 533
Matter.js library, 309–315
soft-body simulations, 343
vehicles and steering, 223
shift() function, 179
show() method, 4
angular motion, 123, 128
fluid resistance, 96
force-directed graphs, 352–353
motion with vectors, 56
oscillation, 142
particle systems, 169, 194–195
pendulums, 159
soft-body simulations, 343
Sierpiński, Wacław, 366
Sierpiński triangles, 366–367, 373
signal processing, 499
Silverman, Brian, 385
simple harmonic motion, 135, 137–140
Sims, Karl, 482–484
sine
coordinate conversion, 131, 160
defined, 126
lookup tables, 280–282
oscillation, 134–137
pendulums, 156–157, 159
sine waves, 117, 134, 139, 142–144
sin() function
coordinate conversion, 131
oscillation, 136–139
pendulums, 160
springs, 147
single-event probability, 10
slice() method, 372, 492
smart rockets, 469–482, 486
developing, 470–475
evolving obstacle avoidance, 478
fitness functions, 471–472
genotype/phenotype distinction, 472
global variables, 471
neuroevolutionary steering behaviors, 565–573
population management, 475–478
SodaConstructor game, 342
soft-body simulations, 342–350
defined, 342
particle-spring connections, 342–343
soft-body characters, 346–350
soft pendulums, 343–345
soft sensors, 499
sohcahtoa mnemonic device, 125–126
sort() function, 178
spatial subdivisions, 115, 275–278
splice() method, 176, 178–179, 307
Spring class and objects, 150–153, 346–347, 353
spring constant (*k*), 148
spring forces, 147–154
current length vs. rest length, 147–148
Hooke's law, 147–148
object-oriented structure, 149–152
square() function, 305–306
standard deviation, 13–16
Stanley, Kenneth O., 550
star-nosed moles, 543
states, 360–363
static methods, 64–66
staying within walls behavior, 231–233
steering behaviors, 215–233
action selection, 216–217
arrive behavior, 224–229
fleeing, 223, 261–265
flocking, 268–274
flow-field following, 233–239
locomotion, 216–217
maximum force, 220–221
maximum speed, 219, 225–227
neuroevolution, 565–573
path following, 240
seeking, 217–224, 260–261
staying within walls, 231–233
steering forces, 216–222, 228–229, 232, 258, 265–266, 268, 284, 510–511, 565, 568, 579
vehicles, 215–216
wandering, 229–231
step() method, 4–5
stochastic fractals, 400, 425–427
strict equality (===) operator, 5
strict inequality (!==) operator, 5
strings
L-systems, 427–428
soft-body simulations, 343–346
Stump, Greg, 585
subclasses, 188–192
sub() method, 45, 47, 66–67
superclasses, 188, 190
super() function, 189–192, 194
supervised learning, 501, 509, 521–523, 549
survival of the fittest/reproducers, 10, 442, 487
synthetic data, 512, 527

T
tangent, 126–128
tension, 154–155
TensorFlow.js, 502
model evaluation, 534–536
parameter tuning, 537
Visor interface, 534–536
testing datasets, 521, 536–537
That Creative Code Page, xxx
thi.ng umbrella, 333
thi.ng vectors, 333
think() method, 555, 557, 579
this keyword, 4, 56

this.x and this.y variables, 305
Thompson, John Henry, xxvii
Thorp, Jer, 469–470
time-series prediction, 499
torque (τ), 124, 158
Toxiclibs.js library, 288
    attraction and repulsion behaviors, 354–357
    compared to Matter.js, 334–335
    force-directed graphs, 351–354
    soft-body simulations, 342–350
    Verlet physics, 333–342
training datasets, 516–517, 521, 524, 532–533, 536
train() method, 533–534
transfer learning, 502
translate() function, 48, 306, 420–421, 432
trapezoid() method, 309
tree structure, 399, 419–427
    deterministic structure, 419–425
    stochastic structure, 425–427
turtle graphics, 432
Turtles, Termites, and Traffic Jams (Resnick), 215
two-dimensional Perlin noise, 25–30
TWO_PI constant, 119, 137, 566

U
Ulam, Stanisław, 361
uniform distributions, 8–9, 19–20
units of measurement, 80
unit vectors, 53, 248
    gravitational attraction, 101–102
    vector normalization, 67
    velocity unit vector, 89–90, 96
universal gravitational constant (*G*), 101–103, 107
unlock() method, 340
unsupervised learning, 501
Unwin, Antony, 525
update() method
    acceleration, 60, 62
    angular motion, 123–124
    force accumulation, 78
    motion with vectors, 56
    neuroevolution, 550
    neuroevolutionary steering behaviors, 570
    Newton's first law, 73
    Newton's second law, 77
    particle systems, 169, 187
    pendulums, 159
    physics engines, 304
    Verlet physics, 336

V
validation datasets, 536–537
variation, 442–443, 445–447, 449, 557–559
Vec2D class and objects, 335–336, 338
Vec3D class and objects, 335
Vector class and objects, 39, 42, 325, 335
vectors, xxxi, 33–69
    acceleration, 59–69
    bouncing ball sketch, 34–36, 43
    components of, 41
    creating, 294
    defined, 34, 37
    drawing, 34
    formulas, 89
    magnitude, 102–103
    motion, 55–59
    normalizing, 102–103
    operations with, 40–55, 295
    p5.js and, 37–40
    position, 38–39, 44
    properties and variables, 36
    scalars vs., 40
    scaling, 48, 102, 295
    vector-based algorithm for motion, 38, 40, 42
    velocity, 38–39, 44
Vehicle class and objects, 215–216, 218–223, 239, 250–252, 259–262, 266, 468
Vehicles (Braitenberg), 216
VerletConstrainedSpring2D class and objects, 339
Verlet integration and physics, 333–342
VerletMinDistanceSpring2D class and objects, 339
VerletParticle2D class and objects, 337–339, 355
VerletSpring2D class and objects, 339, 347, 353
vertex() function, 311, 344
viscous force, 94. *See also* fluid resistance
Visor interface, 534–536
von Neumann, John, 361

W
wagon wheel effect, 124
Walker class and objects, 3–9
    custom distributions, 16–17
    data for, 3–4
    four possible steps, 4–6
    nine possible steps, 7
    nonuniform distributions, 12–13
    Perlin noise, 23–24, 28
wavelength, 143–144
waves, 142–146
    motion, 144–146
    sine waves, 117, 134, 139, 142–144
    static, 143–144
    wavelength, 143–144
WEBGL mode, 210
WebGL (Web Graphics Library) renderer, 210, 274
weight, mass vs., 76
weightedSelection() function, 462, 475, 561
while loops, 405–406, 455
whileTraining() callback function, 534
Windmill class and objects, 320–321
Wolfram, Stephen, 361, 362–363, 367–368, 379
Wolfram Physics Project, 361
World class and objects, 488–489, 492–493

X
XOR (exclusive or), 518–520