

Creating Intelligent Behavior in Games



AI for Game Developers

O'REILLY®

David M. Bourg and Glenn Seemann

A

- A* algorithm, 126, 148
- A* operations, 148
- A* pathfinding, 126–148
- A* pseudo code
 - code, 129
- Activation functions
 - neural network, 278–281
- Adaptability, 231–232
- Adding ants
 - Ant Example code, 176
- Adding fitness tracking
 - code, 338–339
- Adjacent tiles, 131f
 - looking for, 134
- AdjustWeights method
 - code, 298
- Ahead membership function, 207
- AI. (see Artificial intelligence (AI))
- AI Application Programming, 164,
 - 315, 347
- AIDemo2-1
 - downloadable program, 8
- AIDemo2-2, 39, 83, 86
- AIDemo3-2
 - downloadable program, 39
- AIDemo4
 - downloadable program,
 - 55, 57
- AIDemo5-1, 87, 89
- Ai_Entity
 - Ant Example code, 175–176
- Ai_Entity class
 - code, 103, 172–173
 - functions
 - Ant Example code, 178
- Ai_Entity::Forage(void)
 - Ant Example code, 178–179
- Ai_Entity::GoHome(void)
 - Ant Example code, 181–182
- Ai_Entity::Thirsty(void)
 - Ant Example code, 184–185
- AI Game Programming Wisdom, 148, 164,
 - 348
- AI Techniques for Game Programming, 348
- Algorithm
 - A*, 126, 148
 - alternate line vs. Bresenham line, 12
 - basic pathfinding
 - code, 96–97
 - Bresenham, 11–13, 15, 30
 - code, 14–15
 - modified line-of-sight, 31
 - chase
 - code, 7–8
 - evade
 - code, 8
 - flocking, 52
 - genetic, 316–358
 - intercept, 21
 - line of sight
 - chase, 19–20
 - drawbacks, 20
 - steering force activation, 16
 - pathfinding, 129
 - pattern movement, 28–30
 - random movement obstacle avoidance
 - code, 98–99

- Algorithm (*continued*)
 - Rete, 226
 - search, 127–128
 - standard line, 11
- A-life techniques, 4–5
- Alignment, 53, 68–69
- Alignment rule
 - code, 69–70
- Alternate line algorithm
 - vs. Bresenham line algorithm, 12
- Alternative attack network, 259
- Alternative model, 255–256, 256
- Ambush technique, 147
- Ant Example, 170–189
 - ai_Entity, 172
 - ant states, 171
 - defining simulation world, 173–174
 - finite state machine classes and structures, 172–173
 - finite state machine diagram, 171
 - forage function, 178–180
 - populating world, 174–177
 - results, 185–186
 - tiled environment, 172
 - updating world, 177
- Ant states
 - code, 171
- Ant world, 175
- Arbitrary pattern, 51
- Armor responses
 - code, 335–337
- Arrays
 - pattern movement algorithm, 28
 - Troll Settings.txt
 - store creature attributes, 152
- Artificial intelligence (AI), 1
 - defining, 2
 - deterministic vs. nondeterministic, 3
 - entity
 - Ant Example code, 174–175
 - technique, 4
- Artificial life (A-life) techniques, 4–5
- Ask function, 162
- Assertions, 214
- Attack decision neural network, 274
- Attack mode network, 259
- Attack or flee, 273–274
- Attack responses
 - code, 333–335
- Attributes
 - basic script
 - for troll file code, 151–153
 - conditional script
 - code, 151
 - used to set game
 - basic script, 150
- B**
- BackPropagate method
 - code, 302
- Back-propagation training, 285–289
 - adjusting weights, 287–288
 - computing error, 286–287
 - momentum, 288–289
- Back-stepping, 132
- Backward chaining, 217–218
- Basic behavior script
 - code, 154
- Basic pathfinding, 96–98
- Basic pathfinding algorithm
 - code, 96–97
- Basic probability, 228–243
- Basic script
 - code, 150
 - to set attributes
 - for troll file code, 151–153
 - use to set game attributes, 150
- Basic script parsing, 151–154
- Basic state machine model, 165–167
- Basic tracing, 100
- Basic “What is your name?” script
 - code, 159
- Bayesian Artificial Intelligence, 267
- Bayesian classifier, 248
- Bayesian Inference and Decision, 267
- Bayesian network, 245–248
 - example, 246
 - inference, 247–249
 - structure, 245–246
- Bayesian resources
 - Internet, 267
- Bayesian techniques, 244–268
- Bayes’ rule, 243
- Behavior script
 - code, 155
- Benevolent AI script
 - code, 159
- Benevolent computer-controlled characters, 159
- Bias, 274
- Body-fixed coordinate system, 18

- Boids, 53
- Bookkeeping, 265–266
- Boolean logic
 - defining points, 189
- Boolean logic membership function, 194
- Boolean variables
 - global, 154
- Boxed in, 90
- Branching technique, 143–145
- Breadcrumb pathfinding, 101–109
- Breadcrumb trail, 101
- Bresenham algorithm, 15
- Bresenham line algorithm, 11–13, 30
 - vs. alternate line algorithm, 12
 - code, 14–15
- Bresenham tile-based chase, 15
- Buckland, Mat, 348
- Building path, 122
- BuildPath To Target function
 - code, 13
- By air or land, 258
- C**
- CalculateErrors method
 - code, 297, 303
- CalculateNeuronValues method
 - example, 296–297
- Causal chain, 248
- Changing states
 - code, 169
- Character abilities, 229–230
- Character class ability, 230
- Characteristics functions, 193
- Character movement
 - AI script, 155
- Chase algorithm
 - code, 7–8
- Chase/evade demo UpdateSimulation
 - code, 83–84
- Chasing and evading, 6–26, 83–87
 - basic, 7–9
 - with brains, 305–315
 - initialization and training, 306–310
 - learning, 310–315
- Cheating, 4
- Choosing direction
 - code, 113
- Chromosomes
 - randomly assigning
 - code, 332–333
- Classical probability, 232–233
- Classic flocking, 53–55
- Classification
 - fuzzy logic, 191–192
- CleanUp method
 - code, 294–295, 300
- Closed list, 130
 - starting tile, 131
- Closing velocity, 22
- Cohesion, 53, 66–68
 - rule
 - code, 67
- Collision detection, 1
- Common cause network, 248
- Common effect network, 248
- Completed node table, 123
- Completed path, 140
- Computing error
 - back-propagation training, 286–287
- Conditional probability, 242–243
- Conditional probability table
 - example, 246
 - for Player Wins, 260
 - example, 261
 - for strikes thrown, 263
- Conditional script
 - to set attributes
 - code, 151
- Conditions, 230
 - code, 321–322
- Conflict resolution, 217
- Conjugate, 354–355
- Conjunction (logical AND), 200
 - definition, 200
- Constant
 - global
 - using, 168–169
- Continuous environment, 7, 10
 - line of sight chase, 20
- Continuous environment node placement, 146
- Continuous environments
 - line of sight chasing, 15–20
- Control
 - fuzzy logic
 - real-world control applications, 190–191
- Control data structure
 - pattern movement
 - code, 40
- Control example, 205–207

- Control instruction
 - pattern movement algorithm, 28
- Control instructions data structure
 - pattern movement
 - code, 28
- Control structures, 39–42
- CreateIndividual
 - defining
 - code, 331
- Creature slates, 231
- Crisp data, 192
- Crossover, 317, 340
 - adding
 - code, 341
- C value, 134

D

- Data encoding
 - genetics in game development, 328–347
- Decisions under uncertainty, 244–268
- Defuzzification, 193, 203–205
 - code, 210
- Dendrites, 269
- Destination
 - placed open list, 139
- Destination tile, 132, 134, 139
- Determining probabilities, 251
- Deterministic AI
 - vs. nondeterministic, 3
 - techniques, 3
- Deterministic behavior
 - definition, 3
- Diagnostic reasoning, 248
- Dice rolling
 - probabilities, 233
- Direction analysis
 - code, 111–112
- Direction angles, 351
- Disjunction (logical OR), 200
- Dissecting neural networks, 274
- DoAttractCraft2
 - code, 84–85
- Door sound script, 163
- DoPattern function
 - code, 47–50
- DoUnitAI initialization
 - code, 59–60
- Dropping a breadcrumb
 - code, 104–105

- DumpData method
 - code, 304–305

E

- Earth-fixed coordinate system, 18
- Empty node table, 122
- Encoded instructions
 - pattern movement algorithm, 28
- Encoding, 318
 - code, 321
- Encoding structure
 - genetics in game development, 330
- Entity
 - AI
 - Ant Example code, 174–175
- EntityList element
 - Ant Example, 177
- Established game AI, 4–5
- Evade algorithm
 - code, 8
- Evolution, 320
 - genetics in game development, 340–347
- Evolutionary process, 317–318
- Evolving plant life, 321–322
- Examining tile, 134–137
- Example, 138–139
- Expectation, 235
- Expert systems, 212
- Explaining away, 248

F

- Facing player's right, 117
- FeedForward method
 - code, 301–302
- Feed-forward neural network
 - three-layer, 275
- Field of view
 - angle factor, 63
 - bounding lines, 63
 - checks, 62
 - limited, 64
 - narrow, 64
 - wide, 62–63
 - wide check
 - code, 62
- Fighting game strike prediction, 218–219
- Filling node table, 123
- Finding breadcrumbs
 - code, 106–107

- Finding path, 124
- Finite state machine, 4, 165–187
 - definition, 165
 - design
 - ant example, 170, 187
 - behavior and transition functions, 169
 - diagram
 - Ant Example, 171
 - ghost diagram, 166
 - Internet resources, 187
- Finite state machine classes and structures
 - Ant Example, 172–173
- Finite state machine design, 168–170
 - structures and classes, 168–169
- First flower generation, 322–323
 - code, 322–323
- First generation, 318–319
- Floating-point variables, 10
- Flocking, 52–79
- Flocking algorithm, 52
- Flocking example, 55–56
- Flower data encoding, 321–322
- Flower evolution, 320, 325–327
 - code, 325–326
- Flower fitness function
 - code, 323–324
- Flower population, 327
- Flower selection, 320
- Following breadcrumb
 - code, 105
- Following road, 110
- Following shortest path, 108
- Follow the leader, 76–79
- Food, water, and poison regulation
 - Ant Example code, 186
- Forage function
 - Ant Example, 178–180
- Force calculation, 89, 94
- Force size fuzzy sets, 208–209
- Force size variables
 - code, 209
- Forward chaining, 217
- Frequency interpretation, 234
- Functions. (see also Membership functions; Potential functions)
 - activation
 - neural network, 278–281
 - ai.Entity class, 178
 - Ant Example, 175
 - code, 178
 - forage, 178–180
 - GoHome, 180–183
 - thirsty, 183–185
 - Ask, 162
 - BuildPath To Target
 - code, 13
 - characteristics, 193
 - DoPattern
 - code, 47–50
 - flower fitness
 - code, 323–324
 - GoHome
 - Ant Example, 180–183
 - hedge, 200
 - hyperbolic activation tangent, 280
 - initialize
 - code, 222
 - InitializePatternTracking
 - code, 45
 - input variable membership, 202
 - intercept
 - code, 22–23
 - Lenard-Jones potential, 81–82
 - linear activation, 281
 - line of sight chase, 17
 - logistic activation, 278
 - modified UpdateSimulation
 - code, 310–312
 - noisy, 284
 - NormalizePattern
 - code, 33–34
 - predefuzzified output, 204
 - ProcessMove
 - code, 223–225
 - random mutation
 - code, 342–347
 - ReTrainTheBrain
 - code, 313
 - simulation
 - code, 310–312
 - standard C
 - generating random numbers, 228
 - step activation, 279
 - TrainTheBrain
 - code, 308
 - transition
 - finite state machine design, 169
 - game AI code, 169
 - UpdateSimulation
 - code, 45–46, 57–59
 - vector, 355
- Fuzzification, 193–200

- Fuzzification of range
 - and force size variables
 - code, 209
- Fuzzification process, 192
- Fuzzification results, 209
- Fuzzy axioms, 200–201
- Fuzzy data, 192
- Fuzzy input
 - logical rules, 200
- Fuzzy logic, 4, 188–211, 253–254
 - applications, 190
 - basics, 192–205
 - classification, 191–192
 - control, 190–191
 - applications, 189–190
 - definition, 188
 - game AI, 190–192
 - mapping process, 192
 - real-world control applications, 190–191
 - tools
 - third-party, 201
- Fuzzy membership functions
 - code, 198–199
- Fuzzy process overview, 192
- Fuzzy rules, 200–203
 - nested and non-nested
 - code, 210
- Fuzzy sets, 195
 - membership, 192
 - theory, 188
- Fuzzy state machines, 4

G

- Galaga, 27
- Game AI
 - established, 4–5
 - future, 5
 - fuzzy logic, 190–192
 - interpretation of, 1
 - structure
 - code, 168
 - transition functions
 - code, 169
- Game environment
 - includes, 142
- Game Programming Gems, 148
- Generic finite state machine diagram, 166
- Genetic algorithm, 316–358
- Genetics in game development, 327–347
 - behaviors, 330

- data encoding, 328–347
- encoding structure, 330
- evolution, 340–347
- first generation, 330–331
- ranking fitness, 337–338
- role playing example, 328
- scenarios, 329
- selection, 338–340
- GetMaxOutputID method
 - code, 302–303
- GetOutput method
 - code, 301
- Ghost behavior
 - code, 167
- Ghost finite state machine diagram, 166
- Giant taunt, 158
 - script
 - code, 158–159
- Global boolean variables, 154
- Global constant
 - using, 168–169
- Global (earth-fixed) coordinate
 - system, 18
- Global variable
 - example, 306
- Global variables
 - code, 221, 265–266
- GoHome function
 - Ant Example, 180–183
 - code, 181
 - variable declarations, 182
- Grade membership function, 194

H

- Hedge functions, 200
- Heuristic, 132
- Hidden layer, 274
- Hit probabilities, 229
- Hit probability tables, 229
- H value, 133
- Hyperbolic activation tangent function, 280
- Hypothetical arrays
 - Troll Settings.txt
 - store creature attributes, 152

I

- Improved tracing, 100
- Inferences, 256–257
 - Bayesian network, 247–249
- Influence mapping, 146–148

- definition, 146
- records number of kills, 147–148
- use, 147
- Initial flower population, 324
- Initialization, 221–223
 - chasing and evading with brains, 306–310
 - code, 307
- Initialize function
 - code, 222
- Initialize method
 - code, 299–300
- InitializePatternTracking function
 - code, 45
- Initializing world
 - Ant Example code, 173–174
- Initial tile path scores, 133
- Input layer, 274
- Input variable membership functions, 202
- Intelligence attribute
 - modifying, 154
- Intelligent behavior
 - and verbal responses, 157
- Intercept algorithm, 21
- Intercept function
 - code, 22–23
- Intercepting, 20–26
 - and leaders chasing
 - code, 78–79
 - scenario, 23–24
 - corrective action, 25
 - initial trajectories, 24
 - interception, 24–25

K

- Keywords
 - searching for
 - code, 161–162
- Keyword scripting
 - code, 161
- Korb, Kevin, 267
- Kung Fu fighting, 262

L

- Labeling nodes, 121
- Language parser
 - create, 161
- Layers
 - connections, 290
- Leader check
 - code, 77

- Leaders chasing and intercepting
 - code, 78–79
- Lefthanded movement
 - code, 117–119
- Lenard-Jones potential function, 81–82
- Limited-field-of-view check
 - code, 64–65
- Linear activation function, 281
- Line drawing
 - in pixel-based environment, 12
- Line of sight
 - algorithm
 - drawbacks, 20
 - steering force activation, 16
 - calculating from predator to prey, 18
 - chase, 9–10
 - algorithm, 19–20
 - code, 17
 - in continuous environments, 15–20
 - function, 17
 - vs. simple chase, 12
 - tiled environments, 54
 - in tiled environments, 10–15
 - path movement, 97
 - tracing with, 101
- Line segments
 - calculation, 31
 - code, 31–32
- Lists
 - pattern movement algorithm, 28
- Local (body-fixed) coordinate system, 18
- Locked conditional probabilities, 250, 255
- Logical AND, 200
- Logical NOT, 200
- Logical OR, 200
- Logical rules
 - fuzzy input, 200
- Logic practitioners
 - traditional, 190
- Logistic activation function, 278
- Lowest-cost path, 143, 145
 - calculating, 144
- Lua, 149

M

- Magnitude, 350–351
- Making inferences, 251–253
- Mapping process
 - fuzzy logic, 192
- Massively multiplayer online role-playing game (MMORG), 149, 164

- Masters, Timothy, 190
 - Membership calculation results, 207
 - Membership functions, 193–199, 207
 - Boolean logic, 193–194
 - fuzzy
 - code, 198–199
 - grade, 194
 - input variable, 202
 - output, 204
 - reverse grade, 196–197, 352
 - singleton output, 204
 - trapezoid, 197
 - trapped, 253
 - triangular, 195–196
 - Merlin, 160
 - Minimum steering force clipping, 49
 - MMORG, 149, 164
 - Model, 258–259, 262
 - Modified Bresenham line-of-sight
 - algorithm, 31
 - Modified UpdateSimulation function
 - code, 310–312
 - Momentum
 - back-propagation training, 288–289
 - Motion
 - calculation, 16
 - Multilayer feed-forward network, 271
 - Mutually exclusive events, 240
- N**
- Narrow-field-of-view check
 - code, 65
 - Negation (logical NOT), 200
 - definition, 200
 - Neighbors, 57–66
 - average position and heading
 - example, 66
 - code, 61–62
 - position summation
 - code, 66–67
 - separation
 - code, 71–72
 - Networks
 - simple, 248
 - Neural network, 269–315
 - activation functions, 278–281
 - advantages, 270
 - bias, 281–282
 - class, 299–305
 - code, 299
 - control, 271–272
 - hidden layer, 283–285
 - input, 275–277
 - output, 282–283
 - robot control
 - example, 272
 - source code, 289–305
 - layer class, 289–298
 - structure, 274–275
 - threat assessment, 272–273
 - example, 273
 - three-layer feed-forward, 275
 - training, 285
 - weights, 277–278
 - NeuralNetworkLayer
 - class members, 290–292
 - constructor
 - code, 292–294
 - Neural Networks for Pattern Recognition, 315
 - Neuron, 270
 - New ai_Entity
 - Ant Example code, 175–176
 - New function
 - Ant Example, 175
 - New global variable
 - example, 306
 - Nicholson, Ann, 267
 - Nodes
 - search, 127–128
 - Nodes and tiles, 129
 - Noisy function, 284
 - Nondeterministic behavior
 - definition, 3
 - Nondeterministic methods, 3
 - Nonmutually exclusive events, 241
 - Nonplayer characters (NPC), 244
 - Normalize, 351–352
 - NormalizePattern function
 - code, 33–34
 - NPC, 244
 - Numerical example, 257–258, 261–262
- O**
- Obstacle avoidance, 73–76, 87–89
 - code, 75, 88–89
 - Odds, 235
 - Open list, 129
 - Optimizing suggestions, 94–95
 - Output function
 - predefuzzified, 204
 - Output fuzzy sets, 204
 - Output layer, 274
 - Output membership function, 204

P

- Parent tile, 130
 - linking to, 132
- Parsing
 - basic script, 151–154
- Path
 - array initialization
 - code, 30
 - building, 122
 - completed, 140
 - direction calculation
 - code, 14
 - finding, 124
 - following, 109–115
 - following shortest, 108
 - initialization code, 13–14
 - initial tile scores, 133
 - lowest-cost, 143, 145
 - calculating, 144
 - movement
 - line of sight, 97
 - over terrain elements,
 - 144
 - road, 115
 - score calculating, 133
 - shortest vs. quickest,
 - 143–145
 - simple movement, 97
 - square, 49
 - wall-tracing, 120
 - zigzag, 50
- Pathfinding, 1, 4
 - A*, 126–148
 - algorithm, 129
 - basic, 96–98
 - basic algorithm
 - code, 96–97
 - breadcrumb, 101–109
 - workhorse method, 148
- Patrolling pattern
 - code
 - complex, 35
 - simple, 34
- Pattern(s)
 - arbitrary, 51
 - rectangular
 - code, 32–33
 - movement, 33
 - square, 50
 - tile
 - complex movement, 35
 - zigzag, 50
- Pattern array
 - code
 - declarations, 42
 - processing, 29
- Pattern definition, 42–45
- Pattern execution, 45–46
- Pattern function
 - code, 33–34
- Pattern initialization
 - code, 28–29
 - square patrol
 - code, 42–43
 - zigzag
 - code, 44
- Pattern matrix
 - following
 - code, 37–38
 - initialization
 - code, 36
- Pattern movement, 27–51
 - algorithm, 28–30
 - arrays, 28
 - control instruction, 28
 - encoded instructions, 28
 - lists, 28
 - control data structure
 - code, 28, 40
 - physically stimulated environments, 38–39
 - rectangular, 33
 - scripted, 156
 - code, 156
 - stimulated environments, 38–39
 - tiled environments, 30–38
 - tracking, 37
- Pattern results, 50–51
- Pattern setup
 - code, 36
- Pearl, Judea, 267
- Physics for Game Developers, 16, 72
- Pixel-based environment
 - line drawing, 12
- Placing nodes, 121
- Player input
 - code, 160
- Point-slope equation
 - for straight line, 195
- Population
 - defining
 - code, 331
- Population explosion
 - Ant Example, 186
- Possible directions, 111

- Posterior probabilities, 247
 - Potential chase and evade, 86
 - Potential functions
 - based movement, 80–95
 - defined, 81–82
 - for game AI, 80–83
 - Lenard-Jones, 81–82
 - Potential opponent
 - challenging, 155
 - Practical Neural Network Recipes in C++, 190
 - Practical Neural Networks, 315
 - Predefuzzified output function, 204
 - Prediction
 - making, 266–267
 - Predictive reasoning, 248
 - Prey
 - current velocity, 21
 - evasive maneuvers, 24
 - Priest rule example
 - code, 215
 - Prior probability tables, 247
 - Probabilistic-OR, 201
 - Probabilistic Reasoning in Intelligent Systems, 267
 - Probabilities
 - conditional
 - trapped, 255
 - rolling dice, 233
 - trapped, 250
 - Probability
 - calculating, 259–263
 - defined, 232
 - of failure, 232
 - number of players
 - in tavern each evening, 236
 - rules, 238–242
 - subjective, 234
 - techniques for assigning, 236–238
 - of success, 232
 - Problems with obstacles, 98
 - ProcessMove function
 - code, 223–225
 - Python, 149
- R**
- RandomizeWeight method
 - code, 295–296
 - Random movement, 99
 - obstacle avoidance, 98–99
 - code, 98–99
 - Random mutation, 317
 - function
 - code, 342–347
 - Randomness, 228–229
 - Range fuzzy sets, 208–209
 - Range to close, 22
 - Ranking fitness, 319
 - Ranking flower fitness, 323–324
 - Recording player positions
 - code, 103–104
 - Rectangular pattern
 - code, 32–33
 - movement, 33
 - Relative directions, 119
 - Relative heading fuzzy sets, 206
 - Relative heading membership calculations
 - code, 206
 - Rete algorithm, 226
 - ReTrainTheBrain function
 - code, 313
 - Reverse grade membership function, 196–197, 352
 - Reynolds, Craig, 52–53
 - RigidBOdy2D class
 - code, 307
 - Road path, 115
 - Robot control neural network
 - example, 272
 - Role playing example
 - genetics in game development, 328
 - Rolling dice
 - probabilities, 233
 - Root nodes, 247
 - Rule(s), 219–221
 - evaluation, 201
 - fired, 214
 - memory, 214
 - triggered, 214
 - Rule-based AI, 212–227
 - Internet sources, 227
 - Rule-based systems
 - basics, 214–216
 - inference in, 216–218
 - Ruleclass
 - code, 220
 - Rule matrix, 209
 - Running simulation
 - Ant Example code, 177
- S**
- Scalar division, 354, 358
 - Scalar multiplication, 354, 357

- Scoring, 132–140
- Scoring with terrain cost, 142
- Script
 - basic, 150–154
 - behavior
 - code, 154
 - benefits, 157
 - benevolent AI
 - code, 159
 - to check player input, 161
 - conditional
 - to set attributes, 151
 - door sound, 163
 - giant taunt
 - code, 158–159
 - in-depth resources, 164
 - keyword
 - code, 161
 - and linking, 163
 - pattern movement
 - code, 156
 - purpose, 149
 - reading data from
 - troll file code, 152–153
 - rules-based systems, 4
 - trap event
 - code, 162
 - triggered sound
 - code, 164
 - verbal taunt
 - code, 157
- Scripted AI, 149–164
- Scripted pattern movement, 156
- Scripting behavior
 - purpose, 154
- Scripting engines, 149–164
- Scripting events, 162–164
- Scripting language, 149
- Scripting opponent attributes, 150–151
- Scripting opponent behavior, 154–157
- Scripting shells
 - Internet sources, 227
- Scripting system, 149
 - add predefined global variables, 154
- Scripting techniques, 149–150
- Scripting verbal interaction, 157–162
- Search
 - algorithm and nodes, 127–128
 - for keywords
 - code, 161–162
 - starting, 128–132
- Search area
 - defining, 126–128
 - limitations, 137
 - simplifying, 127
- Search term
 - string variable, 153–154
- Selection, 319–320
- Separation, 53, 70–73
- SetDesiredOutput method
 - code, 301
- SetInput method
 - code, 300
- SetLearningRate method
 - code, 303
- SetLinearOutput method
 - code, 303–304
- SetMomentum method
 - code, 304
- SetRule method
 - code, 221
- Set units[i] member variables
 - code, 73
- Seven fuzzy sets, 197–198
- Simple networks, 248
- Simulation function
 - chase/evade demo
 - code, 83–84
 - code, 45–46, 57–59, 310–312
 - running
 - Ant Example code, 177
- Simulation world
 - defining
 - Ant Example, 173–174
- Singleton output membership functions, 204
- Singleton output values, 209
- Sorting fitness
 - code, 339–340
- Sound effects
 - triggering, 163
- Square path, 49
- Square patrol pattern initialization
 - code, 42–43
- Square pattern, 50
- Square tile-based games, 9
- Square tiles, 10
- Standard C function
 - generating random numbers, 228
- Standard line algorithm, 11
- State change tracking structure
 - code, 41
- State transitions, 230–231

- Steering force calculation, 74
 - code, 207
 - Steering force test, 19
 - Steering model, 56–57
 - Step activation function, 279
 - Stimulated environments
 - pattern movement, 38–39
 - Straight line
 - equation, 195
 - Strike network, 262
 - Strike prediction, 223–226,
 - 264–265
 - code, 264–265
 - String
 - uppercase vs. lowercase, 162
 - String parameters
 - and spaces in code, 153
 - String variable
 - search term, 153–154
 - Strong AI, 2
 - Subjective interpretation, 234–235
 - Subjective probability, 234
 - techniques for assigning, 236–238
 - Sums of five
 - in roll of two dice, 233
 - Sums of seven
 - in roll of two dice, 233
 - Supervised training, 285
 - S value, 133
 - Swarming, 89–94
 - with chasing and obstacle avoidance
 - code, 92–94
 - code, 90–91
 - Switch statement, 167
- T**
- Team constants
 - code, 172
 - Technology tree example, 213
 - Temple rule example
 - code, 215
 - Terrain
 - hypothetical types, 142
 - types, 142
 - Terrain analysis
 - code, 110
 - Terrain array
 - Ant Example code, 173
 - Terrain cost
 - cost equation, 142
 - Terrain elements
 - adding, 143
 - Terrain type
 - grassland, 142
 - Terrain values
 - Ant Example code, 173
 - Third-party fuzzy logic tools, 201
 - Thirsty function
 - Ant Example, 183
 - code, 184–185
 - variables, 185
 - Threat assessment
 - example, 207–211
 - fuzzy logic, 191
 - neural network
 - example, 273
 - Three-layer feed-forward neural network,
 - 275
 - Three-node chain, 255
 - Three nonmutually exclusive events, 241
 - Thrust factor, 48
 - Tile(s)
 - linking together, 130
 - and nodes, 129
 - vs. nodes, 128
 - Tile-based chase, 9
 - code, 8
 - Tile-based eight-way movement, 11
 - Tile-based games, 7, 109
 - Tiled environments, 128–129
 - Ant Example, 172
 - line of sight chase, 10–15, 54
 - pattern movement, 30–38
 - Tiled search area, 128
 - creating, 130
 - Tile patterns
 - complex movement, 35
 - Tile values
 - calculating, 135
 - Time to close, 22
 - Tracing around obstacles, 99–101
 - Tracing with line of sight, 101
 - Tracking
 - pattern movement, 37
 - Tracking hit-point damage
 - code, 337–338
 - Trail array initialization
 - code, 103
 - Training
 - chasing and evading with brains,
 - 306–310
 - neural network, 285
 - TrainTheBrain function
 - code, 308

- Transition functions
 - finite state machine design, 169
 - game AI
 - code, 169
 - state, 230–231
- Trap event script
 - code, 162
- Trapezoid membership function, 197
- Trapped, 249–250
 - conditional probabilities, 255
 - membership functions, 253
 - probabilities, 250
- Treasure, 254–255
- Tree diagram, 250–251
- Triangular membership function, 195–196
- Triggered sound script
 - code, 164
- Triple scalar product, 358
- Troll Settings.txt, 151
 - hypothetical arrays
 - store creature attributes, 152
- Two-node chain, 250

U

- Unit awareness
 - of local surroundings, 54
- Unit field
 - of view, 55
- Unit radius, 54
- Unit visibility, 54
- Unsupervised training, 285
- Update position
 - code, 114
- UpdateSimulation function
 - code, 45–46, 57–59

V

- Variable(s)
 - declarations
 - GoHome function, 182
 - forage function
 - Ant Example, 179–180
 - global
 - code, 221, 265–266
 - new global
 - example, 306

- Vector addition, 353, 355
- Vector calculation, 74
- Vector class, 349–350
- Vector cross product, 355–356
- Vector dot product, 356–357
- Vector functions, 355
- Vector length, 350
- Vector operations, 349–357
- Vector record, 352
- Vector subtraction, 353, 355
- Vehicle forces, 17
- Venn diagram, 239
- Verbal taunt script
 - code, 157
- Virtual feelers, 73
- Visibility models, 62

W

- Wall tracing, 115–120
- Wall-tracing path, 120
- Waypoint navigation, 120–125
- Weak AI, 1, 2
- Weighting directions, 113
- Weights
 - adjusting
 - back-propagation training, 287–288
 - neural network, 277–278
- Wide field of view, 62, 63
- Wide field of view check
 - code, 62
- Winkler, Robert, 267
- Woehr, Jack, 188
- Woodcock, Steven, 1, 148
- Workhorse pathfinding method, 148
- Working memory, 214, 219
- Working memory example
 - code, 214

Z

- Zadeh, Lotfi, 188
- Zigzag path, 50
- Zigzag pattern, 50
- Zigzag pattern initialization
 - code, 44