

Functions

resource::message	1:65
resource::message	1:69
resource::resource	1:73

```

1  #include <iostream>
2  #include <fstream>
3  #include <stdlib.h>
4  #include <math.h>
5  #include <string.h>
6  #include <stdio.h>
7
8  // program paramaters
9  const int WorldSize = 100;
10 const int GenerationLimit = 5000;
11 const int InitialCrittters = 25;
12 const int MaxPopulation = 10000;
13 const float FOOD_ADDITION_FACTOR = 0.02;
14
15 // ages critters start dying of old age
16 const int MiddleAge = 10;
17 const int OldAge = 15;
18
19 const bool VERBOSE = false;
20 const bool BIG_OUTPUT = false;
21 const int MAX_CHROMOSOMES = 10;
22
23 // following are decision return codes
24 const int DECIDE_DIE = -1;
25 const int DECIDE_FEED = -2;
26 const int DECIDE_MOVE = -3;
27 const int DECIDE_REPRODUCE = -4;
28 // +ve number is the food left in the square the critter on after
29 // it has fed
30
31 struct genes {
32     int energy_per_move; // energy used each move
33     int num_offspring; // number of beings this splits into when reproducing
34     int split_energy; // amount of energy needed before reproduction
35     int vision_range; // cells in each direction, so 1 gives 8 cells
36     double move_range; // distance critter can move in 1 turn
37
38     // mutation related stuff
39     int num_chromosomes;
40     int *chromosomes[MAX_CHROMOSOMES];
41     int mutations_factor[MAX_CHROMOSOMES];
42     int mutations_chance[MAX_CHROMOSOMES];
43 };
44
45 // ***** //
46 // ***** RESOURCE CLASS ***** //
47 // ***** //
48
49 class resource {
50 public:
51     resource(void);
52     int getvalue(int xpos, int ypos);
53     void add(int num_to_add);
54     void add(void);
55     void message(char message[100]);
56     void message(char message[100], int num);
57     void setvalue(int xpos, int ypos, int newval);
58     void report(ofstream &file);
59     int getTotalFood(void);
60 private:
61     int contents[WorldSize+1][WorldSize+1]; // the food present in each cell
62     int i,j; // counting constants
63 };
64
65 void resource::message(char message[100]) {
66     if (VERBOSE) { cout << message << "\n"; }
67 }
68
69 void resource::message(char message[100], int num) {
70     if (VERBOSE) { cout << message << num << "\n"; }
71 }
72
73 resource::resource(void) {
74     // constructor, set all values to 0
75     int i,j;

```

Functions	
resource::add	2:91
resource::add	2:109
resource::getvalue	2:83
resource::message	1:65
resource::message	1:69
resource::report	2:140
resource::resource	1:73
resource::setvalue	2:87

```

76     for (i=0;i<=WorldSize;i++) {
77         for (j=0;j<=WorldSize;j++) {
78             contents[i][j]=0;
79         }
80     }
81 }
82
83 int resource::getvalue(int xpos, int ypos) {
84     return contents[xpos][ypos];
85 }
86
87 void resource::setvalue(int xpos, int ypos, int newval) {
88     contents[xpos][ypos] = newval;
89 }
90
91 void resource::add (int num_to_add) {
92     /* menthod to add a 100 units of
93     energy to a specified number of cells */
94
95     int i,j,k; // usual counting variables
96     long int total_food=0; // debugging summation
97
98     for (k=0 ; k<=num_to_add ; k++) {
99         i=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0)); // code from 'man S 3 rand'
100        j=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0));
101        if (contents[i][j] > 155) {
102            contents[i][j]=255;
103        } else {
104            contents[i][j]+=100;
105        }
106    }
107 }
108
109 void resource::add (void) {
110     /* similar method to the above, but adds according to the
111     FOOD_ADDITION_FACTOR program paramater instead of a set
112     number of cells */
113
114     int i,j,k; // usual counting variables
115     long int total_food=0; // debugging summation
116     int num_to_add;
117
118     num_to_add = (int)(WorldSize * WorldSize * FOOD_ADDITION_FACTOR);
119     message("Adding ",num_to_add);
120
121     for (k=0 ; k<=num_to_add ; k++) {
122         i=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0)); // code from 'man S 3 rand'
123         j=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0));
124         if (contents[i][j] > 155) {
125             contents[i][j]=255;
126         } else {
127             contents[i][j]+=100;
128         }
129     }
130
131     for (i=0;i<=WorldSize;i++) {
132         for (j=0;j<=WorldSize;j++) {
133             if ( (contents[i][j]<0) || (contents[i][j]>255) ) {
134                 printf("\t\tFood outside range at index (%4d,%4d)->%10d\n",i,j,contents[i][j]
135             );
136         }
137     }
138 }
139
140 void resource::report(ofstream &file) {
141     int i,j;
142     char info[40];
143
144     for (i=0;i<=WorldSize;i++) {
145         for (j=0;j<=WorldSize;j++) {
146             sprintf(info, "(%3d,%3d): %3d", i, j, contents[i][j]);
147             file << info << "\n";
148         }
149     }

```

```

150     file << "\n"; // blank line to mark space between generations
151 }
152
153 int resource::getTotalFood(void) {
154     int totalFood=0;
155     int i,j;
156     for (i=0;i<=WorldSize;i++) {
157         for (j=0;j<=WorldSize;j++) {
158             totalFood += contents[i][j];
159         }
160     }
161     return totalFood;
162 }
163
164 // *****
165 // ***** CRITTER CLASS *****
166 // *****
167
168
169 class critter {
170     public:
171         // fields
172         genes genome;
173         bool alive; // is this critter alive?
174         int energy; // initial energy
175         int xpos, ypos; // initial x and y positions
176         int age; // age in turns
177         critter(void);
178         // methods
179         int decide(resource foodstuff);
180         void kill(void);
181         void birth(int init_energy);
182         void birth(int init_xpos, int init_ypos, int init_energy, genes parent);
183         int get_chromosome_value(int chromosome);
184         genes reproduce(void);
185         genes mutate(genes baby);
186         void report(ofstream &file, int index);
187     private:
188         // fields
189         int i,j; // counting constants
190         // methods
191         void feed(resource &foodstuff);
192         void move(int x, int y);
193         void message(char message[100]);
194         void message(char message[100],int num);
195 };
196
197 void critter::message(char message[100]) {
198     if (VERBOSE) { cout << message << "\n"; }
199 }
200
201 void critter::message(char message[100],int num) {
202     if (VERBOSE) { cout << message << " " << num << "\n"; }
203 }
204
205 critter::critter(void) {
206     alive=false;
207     xpos=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0)); // code from 'man S 3 rand'
208     ypos=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0));
209 };
210
211 void critter::feed(resource &foodstuff) {
212     int current_value;
213     current_value = foodstuff.getvalue(xpos,ypos);
214     message("\t\tEnergy is",energy);
215     message("\t\tcurrent_value is", current_value);
216     if (current_value > 100) {
217         foodstuff.setvalue(xpos,ypos,current_value-100);
218         energy+=100;
219     } else {
220         foodstuff.setvalue(xpos,ypos,0);
221         energy+=current_value;
222     }
223     message("\t\tEnergy is",energy);
224     message("\t\tcurrent_value is", foodstuff.getvalue(xpos,ypos));

```

Functions

critter::critter	3:205
critter::feed	3:211
critter::message	3:197
critter::message	3:201
resource::add	2:91
resource::add	2:109
resource::getTotalFood	3:153
resource::getvalue	2:83
resource::message	1:65
resource::message	1:69
resource::report	2:140
resource::resource	1:73
resource::setvalue	2:87

```

225 };
226
227 void critter::kill(void) {
228     alive=false;
229 }
230
231 void critter::birth(int init_energy) {
232     /* this function used to birth a normal critter */
233     alive=true;
234     energy = init_energy;
235
236     genome.energy_per_move = 25;
237     genome.num_offspring = 2;
238     genome.split_energy = 200;
239     genome.vision_range = 1;
240     genome.move_range = 1;
241     genome.move_range = 1.5; // one cell, or one diagonal move
242
243     // mutation stuff
244     genome.num_chromosomes = 1;
245     genome.chromosomes[0] = & genome.num_offspring;
246     genome.mutations_factor[0]=1;
247     genome.mutations_chance[0]=2; // %age chance, ie 50 == half the time
248
249     xpos=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0)); // code from 'man S 3 rand'
250     ypos=1+(int) ((double)WorldSize*rand()/(RAND_MAX+1.0));
251
252     age=1;
253 }
254
255 void critter::birth(int init_xpos, int init_ypos, int init_energy, genes parent) {
256     int randFactor;
257
258     /* this function used to birth a child of an existing critter */
259     alive=true;
260     energy=init_energy;
261
262     randFactor=-1+(int) (3.0*rand()/(RAND_MAX+1.0));
263     xpos= ((init_xpos+randFactor) % WorldSize);
264     randFactor=-1+(int) (3.0*rand()/(RAND_MAX+1.0));
265     ypos= ((init_ypos+randFactor) % WorldSize);
266
267     genome = parent;
268     age=1;
269 }
270
271 int critter::decide(resource foodstuff) {
272     /* this function is where the critter decides what to do
273     * each timestep. We step through each possibility and
274     * calculate a weighting for it, then do the highest
275     * weighted choice. */
276
277     if (alive==false) { return -255; }
278
279     energy -= genome.energy_per_move;
280
281     int move_weights[WorldSize][WorldSize];
282     int max_move_weight(0);
283     int max_move_weight_location[1];
284     int feed_weight;
285     int chance;
286
287     // first, weigh move choices for each square in the vision range
288     for (i=((xpos-genome.vision_range)%(WorldSize-1)); i<=((xpos+genome.vision_range)%(
WorldSize-1)); i++) {
289         for (j=((ypos-genome.vision_range)%(WorldSize-1)); j<=((ypos+genome.vision_range)
%(WorldSize-1)); j++) {
290             move_weights[i][j]=foodstuff.getvalue(i,j);
291             if (move_weights[i][j] > max_move_weight) {
292                 max_move_weight = move_weights[i][j];
293                 max_move_weight_location[0]=i;
294                 max_move_weight_location[1]=j;
295             }
296         }
297     }

```

Functions

critter::birth	4:231
critter::birth	4:255
critter::critter	3:205
critter::decide	4:271
critter::feed	3:211
critter::kill	4:227
critter::message	3:197
critter::message	3:201
resource::add	2:91
resource::add	2:109
resource::getTotalFood	3:153
resource::getvalue	2:83
resource::message	1:65
resource::message	1:69
resource::report	2:140
resource::resource	1:73
resource::setvalue	2:87

		Functions
298	message (" \tmax feed weight is",max_move_weight);	critter::birth 4:231
299		critter::birth 4:255
300	// feed weight	critter::critter 3:205
301	/* need a slightly arbitrary bit of maths here to decide	critter::decide 4:271
302	* whether to feed from the current square or not */	critter::feed 3:211
303	// possible FIXME	critter::kill 4:227
304	feed_weight = foodstuff.getvalue(xpos,ypos) * (genome.split_energy-e	critter::message 3:197
305	message (" \tfeed weight is",feed_weight);	critter::message 3:201
306	message (" \tenergy is",energy);	critter::move 5:336
307		critter::mutate 5:361
308	// check old age	critter::reproduce 5:357
309	// apply a simple linear chance of death with increasing age	resource::add 2:91
310	chance=MiddleAge+1+(int) ((double)(OldAge - MiddleAge)*rand()/(RAND_	resource::add 2:109
311	if (age>chance) {	resource::getTotalFood 3:153
312	message("Dying - old age at age",age);	resource::getValue 2:83
313	return DECIDE_DIE;	resource::message 1:65
314	}	resource::message 1:69
315		resource::report 2:140
316	// now decide	resource::resource 1:73
317	if (energy <=0) {	resource::setValue 2:87
318	// die	
319	message(" \tdying (starvation)");	
320	return DECIDE_DIE;	
321	} else if (energy>genome.split_energy) {	
322	message (" \treproducing");	
323	// return (energy / genome.num_offspring);	
324	return DECIDE_REPRODUCE;	
325	} else if (feed_weight > max_move_weight) {	
326	message (" \tfeeding");	
327	feed(foodstuff);	
328	return foodstuff.getvalue(xpos,ypos);	
329	} else {	
330	message(" \tmoving");	
331	move (max_move_weight_location[0], max_move_weight_location[1]);	
332	return DECIDE_MOVE;	
333	}	
334	}	
335		
336	void critter::move(int x, int y) {	
337	/* need to handle cases where the critter can see further than	
338	* it can move. In those cases, we move towards the greatest	
339	* amount of food */	
340	// x is the x-location with most food, y the y-location	
341	double dist; // distance to food	
342	double newxpos, newypos; // new position	
343		
344	// first, calc the distance to the square	
345	dist = sqrt((abs(xpos-x)^2 + abs(ypos-y)^2));	
346	if (dist < genome.move_range) {	
347	// critter can get all the way to the food in one turn	
348	xpos=x;	
349	ypos=y;	
350	} else {	
351	// critter can't get all the way there; move part of the way	
352	/* this will only get used if the critter's move and vision distances	
353	are different, which is not the case in this version of the program */	
354	}	
355	}	
356		
357	genes critter::reproduce(void) {	
358	return mutate(genome);	
359	}	
360		
361	genes critter::mutate(genes baby) {	
362	int chromosome; // counter	
363	double chance; // random number	
364		
365	for (chromosome=0;chromosome<=baby.num_chromosomes;chromosome++) {	
366	message("Testing for mutation in chromosome",chromosome);	
367	chance=1+(int) (100.0*rand()/(RAND_MAX+1.0));	
368	if (chance<=baby.mutations_chance[chromosome]) {	
369	// mutation takes place; mutate number up or down?	
370	chance=1+(int) (100.0*rand()/(RAND_MAX+1.0));	
371	if (chance < 50) { // mutate value down	
372	if (*baby.chromosomes[chromosome] - baby.mutations_factor[chromosome] > 0) {	

	Functions
373 *baby.chromosomes[chromosome] -= baby.mutations_factor[chromosome];	critter::birth 4:231
374 message("\tMutation downwards to",*baby.chromosomes[chromosome]);	critter::birth 4:255
375 }	critter::critter 3:205
376 } else { // mutate value up	critter::decide 4:271
377 *baby.chromosomes[chromosome] += baby.mutations_factor[chromosome];	critter::feed 3:211
378 message("\tMutation upwards to",*baby.chromosomes[chromosome]);	critter::get_chromosome 6:403
379 }	critter::kill 4:227
380 }	critter::message 3:197
381 }	critter::message 3:201
382 }	critter::move 5:336
383 return baby;	critter::mutate 5:361
384 }	critter::report 6:386
385 }	critter::reproduce 5:357
386 void critter::report(ofstream &file, int index) {	main 6:416
387 char info[40];	resource::add 2:91
388 }	resource::add 2:109
389 if (alive == false) { return; }	resource::getTotalFood 3:153
390 }	resource::getvalue 2:83
391 // do some output to the critter file	resource::message 1:65
392 sprintf(info,"%5d: (%3d,%3d) %3d %3d",	resource::message 1:69
393 index,xpos,ypos,energy,age);	resource::report 2:140
394 file << info << " / ";	resource::resource 1:73
395 sprintf(info,"%2d %2d %3d %2d %2d",	resource::setvalue 2:87
396 genome.energy_per_move,genome.num_offspring,	
397 genome.split_energy,genome.vision_range,	
398 genome.move_range);	
399 file << info << "\n";	
400 }	
401 }	
402 }	
403 int critter::get_chromosome_value(int chromosome) {	
404 return *genome.chromosomes[chromosome];	
405 }	
406 }	
407 // *****	
408 // ***** main() *****	
409 // *****	
410 }	
411 void end_run();	
412 void message(char message[100]);	
413 void message(char message[100], int num);	
414 void random_seed(void);	
415 }	
416 int main() {	
417 message("Beginning program");	
418 random_seed();	
419 }	
420 }	
421 message("Creating output files");	
422 ofstream critter_out("critter.txt",ios::out ios::trunc);	
423 ofstream resource_out("resource.txt",ios::out ios::trunc);	
424 ofstream geneplot_out("stats.txt",ios::out ios::trunc);	
425 }	
426 message("Creating critters");	
427 critter critters[MaxPopulation];	
428 message("Creating food");	
429 resource food;	
430 int generation,critter,child,i,j; // count variables	
431 int decision; // return value from critter.decision	
432 int child_energy, child_xpos, child_ypos; // used to birth new critters	
433 bool breakloop = false; // used to break a for loop below	
434 int critter_count=1; // count number of critters alive for debugging	
435 }	
436 unsigned long int gene_totals[MAX_CHROMOSOMES]; // total for gene values	
437 int chromosome; // counter	
438 }	
439 genes parent;	
440 message("Adding food");	
441 food.add();	
442 }	
443 for (i=1;i<=InitialCritters;i++) {	
444 critters[i].birth(100);	
445 }	
446 }	
447 if (VERBOSE==false) { printf("Generation: ["); }	


```

519 // do some reporting and housekeeping
520 if (BIG_OUTPUT == true) {
521     food.report(resource_out);
522 }
523
524 geneplot_out << generation << " " << critter_count << " ";
525 geneplot_out << food.getTotalFood() << " ";
526
527 for (chromosome=0;chromosome<MAX_CHROMOSOMES;chromosome++) {
528     geneplot_out << (double(gene_totals[chromosome])/double(critter_
529 )
530 }
531 geneplot_out << "\n";
532
533 message("Critters alive: ",critter_count);
534 if (VERBOSE==false) {printf("\b\b\b\b\b\b\b\b"); }
535 critter_out << "\n";
536 if (critter_count == 0) {
537     // we have extinction
538     critter_out.close();
539     resource_out.close();
540     printf("\nExtinction.\n");
541     end_run();
542 }
543 }
544 }
545
546 void end_run(void) {
547     cout << "\n\nRun complete.\n";
548     exit(1);
549 }
550
551 void message(char message[100]) {
552     if (VERBOSE) { cout << message << "\n"; }
553 }
554
555 void message(char message[100], int num) {
556     if (VERBOSE) { cout << message << num << "\n"; }
557 }
558
559 /* random seed generator function
560 taken from http://sourceforge.net/snippet/detail.php?type=snippet&id=100096
561 as we're using Linux, we can use /dev/random as a nice, big entropy source
562 */
563 void random_seed(void) {
564     FILE *a;
565     int showmeyourseed;
566     a=fopen("/dev/random","r");
567     showmeyourseed=getc(a);
568     srand(showmeyourseed);
569     fflush(a);
570 }
571
572

```

Functions

critter::birth	4:231
critter::birth	4:255
critter::critter	3:205
critter::decide	4:271
critter::feed	3:211
critter::get_chromosom	6:403
critter::kill	4:227
critter::message	3:197
critter::message	3:201
critter::move	5:336
critter::mutate	5:361
critter::report	6:386
critter::reproduce	5:357
end_run	8:546
main	6:416
message	8:551
message	8:555
random_seed	8:563
resource::add	2:91
resource::add	2:109
resource::getTotalFood	3:153
resource::getvalue	2:83
resource::message	1:65
resource::message	1:69
resource::report	2:140
resource::resource	1:73
resource::setvalue	2:87