```
Assignment 2 Beta Testing Script
--------------------------------
[2019/10/27 13:33:02] Beginning test script.
[2019/10/27 13:33:02] Searching for makefile.
[2019/10/27 13:33:02] Makefile found:
./926902Algds/Makefile
[2019/10/27 13:33:03] Moving to directory: ./926902Algds
[2019/10/27 13:33:03] Cleaning up files.
[2019/10/27 13:33:03] Generating and printing code diff.
grep: pacman.c: No such file or directory
[2019/10/27 13:33:03] Using /home/jovyan/base/old/notflat for code diff
[2019/10/27 13:33:03] diff -arw -x '*.o' -x 'COPYING' -x 'ChangeLog' -x 'README' -x 'Levels' -x 'LevelsTest' -x '*.txt' /home/jovyan/base/old/notflat . > _codeDiff.txt
Only in .: .DS_Store
Only in .: Levels
diff -arw -x '*.o' -x '*.txt' /home/jovyan/base/old/notflat/Makefile ./Makefile
2,3c2,3
< CPPFLAGS=-Wall   -O3 -std=gnu99
< #CPPFLAGS=   -g  -std=gnu99
---
> #CPPFLAGS=-Wall   -O3 -std=gnu99
> CPPFLAGS=   -g  -std=gnu99
Only in /home/jovyan/base/old/notflat: pacman.c
Only in /home/jovyan/base/old/notflat: pacman.h
diff -arw -x '*.o' -x '*.txt' /home/jovyan/base/old/notflat/src/ai.c ./src/ai.c
10a11,12
> #define EPSINON 0.00001
>
76a79
>       int i = 0, l = 0, g = 0;
78a82,99
>       if (n->parent!=NULL){
>
>               // Pacman eaten a fruit and Incincible
>               if ((n->parent ->state.Invincible == 0) && (n ->state.Invincible == 1)){
>                       i = 10;
>               }
>
>               // Lost a life
>               if ( (n->parent->state.Lives) - (n->state.Lives) == 1){
>                       l = 10;
>               }
>
>               // Game over
>               if (n->state.Lives < 0){
>                       g = 100;
>               }
>       }
>       h = i - l - g;
86a108,118
>       float value = heuristic(n);
>       float score_n = n->state.Points;
>       float score_nParent;
>
>       // The change in score from the current node and the parent node
>       if (n->parent == NULL){
>               score_nParent = score_n;
>       }
>       else{
>               score_nParent = n->parent->state.Points;
>       }
88c120,121
<       float discount = pow(0.99,n->depth);
---
>       //r(n) = (h(n) + score(n) - score(nParent)) \002
>       reward = value + score_n - score_nParent;
89a123
>       float discount = pow(0.99,n->depth);
98a133
>       *new_node = (node_t *) malloc(sizeof(node_t));
100,101c135,136
<     //FILL IN MISSING CODE
<
---
>       // Updates the state with the action chosen
>       copy_state(&( (*new_node)->state ), &(n->state) );
104c139,145
<       return changed_dir;
---
>       // A new node points to the parent
>       (*new_node)->parent = n;
>       (*new_node)->depth = n->depth+1;
>
>       // Updates the priority with negative depth
>       (*new_node)->priority = -((*new_node)->depth);
>       (*new_node)->num_childs = 0;
```

```
 86: 106c147,148
 87: < }
 88: ---
 89: >       // Pass the reward to the node for the use of record score
 90: >       (*new_node)->acc_reward = get_reward(*new_node);
 91: 107a150,157
 92: >       if ( (*new_node)->depth == 1 ){
 93: >               (*new_node)->move = action;
 94: >       }
 95: >       else{
 96: >               (*new_node)->move = (*new_node)->parent->move;
 97: >       }
 98: >       return changed_dir;
 99: > }
100: 114a165,168
101: >       clock_t begin, end;
102: >     double cost;
103: >     begin = clock();
104: >
105: 125a180,181
106: >       //node <- start
107: >       node_t* n = create_init_node( &init_state );
108: 127,128c183,188
109: <       //Add the initial node
110: <       //node_t* n = create_init_node( &init_state );
111: ---
112: >       //explored <- empty Array
113: >       node_t** explored = malloc(sizeof(node_t*)* budget);
114: >       memset(explored,0,sizeof(node_t*));
115: >
116: >       // frontier <- priority Queue Containing node Only
117: >       explored[generated_nodes] = n;
118: 131c191,259
119: <       //heap_push(&h,n);
120: ---
121: >       //node <- frontier.pop()
122: >       heap_push(&h, n);
123: >       do{
124: >               node_t* h_prior_node = heap_delete(&h);
125: >
126: >               // if size(explored) < budget
127: >               if (expanded_nodes < budget){
128: >                       expanded_nodes++;
129: >
130: >                       // for each APPLICABLE action {Left;Right;Up;Downg}
131: >                       for (int movement=0; movement<4; movement++){
132: >
133: >                               // filter all un-available actions
134: >                               if(applyAction(h_prior_node, &n, movement) == false){
135: >                                       continue;
136: >                                       free(explored);
137: >                               }
138: >
139: >                               /* newNode <- applyAction(node)
140: >                                * simulate the next move
141: >                                */
142: >                               node_t* new_node=NULL;
143: >                               bool changed_dir = applyAction(h_prior_node, &new_node, movement);
144: >
145: >
146: >                               // propagateBackScoreToFirstAction(newNode)
147: >                               if (new_node->depth > max_depth){
148: >                                       max_depth = new_node->depth;
149: >                               }
150: >
151: >                               h_prior_node->num_childs++;
152: >
153: >                               generated_nodes++;
154: >
155: >                               // Memory management
156: >                               if (((sizeof(explored)/sizeof(node_t**)) < generated_nodes)){
157: >                                       explored = (node_t**)realloc(explored, sizeof(node_t*)* 10000);
158: >                               }
159: >
160: >                               explored[generated_nodes] = new_node;
161: >
162: >                               // if lostLife(newNode)
163: >                               if (new_node->state.Lives < h_prior_node->state.Lives){
164: >                                       // node not into queue
165: >                               }
166: >                               else{
167: >                                       // frontier.add(newNode)
168: >                                       heap_push(&h, new_node);
169: >                               }
170: >                       }
171: >
```

```
  172: >                }
  173: >        } while (h.count > 0);
  174: >
  175: >        // Calculte the number of items in the array
  176: >        int length = 0;
  177: >        while(explored[length]!=NULL){
  178: >                length++;
  179: >        }
  180: >
  181: >        // Calculate the max and avg by looping start at the leaves node backward
  182: >        for( int explore_index = generated_nodes; explore_index>-1; explore_index-- ){
  183: >
  184: >                node_t* tmp_node = explored[explore_index];
  185: >
  186: >                /* Find the childs location
  187: >                 * Add the points reward
  188: >                 */
  189: >                int counter = 0;
  190: 133c261,289
  191: <        //FILL IN THE GRAPH ALGORITHM
  192: ---
  193: >                // The deepest child node is found
  194: >                if (tmp_node->num_childs==0){
  195: >                        float leaf_acc_reward = 0;
  196: >                        leaf_acc_reward += tmp_node->acc_reward;
  197: >
  198: >                        // Add the points from the parent all the path
  199: >                        node_t* p_tem_node =tmp_node->parent;
  200: >                        do{
  201: >                                leaf_acc_reward += p_tem_node->acc_reward;
  202: >                                p_tem_node = p_tem_node->parent;
  203: >                                counter++;
  204: >                        } while (p_tem_node!=NULL);
  205: >
  206: >                        // Propagation Max
  207: >                        if (propagation == 0){
  208: >                                if (leaf_acc_reward > best_action_score[tmp_node->move]){
  209: >                                        best_action_score[tmp_node->move] = leaf_acc_reward;
  210: >                                }
  211: >                        }
  212: >
  213: >                        // Propagation Average
  214: >                        if (propagation == 1){
  215: >                                if ( (leaf_acc_reward/counter) > (best_action_score[tmp_node->move]/coun
ter)){
  216: >                                        best_action_score[tmp_node->move] = (leaf_acc_reward/counter);
  217: >                                }
  218: >                        }
  219: >
  220: >                }
  221: >        }
  222: 134a291,341
  223: >        // FreeMemory
  224: >        emptyPQ(&h);
  225: >
  226: >        // Find the highest score and break the tie
  227: >        int maxIndex = 0;
  228: >        int tie = 0;
  229: >        double max = 0;
  230: >        for (int i = 0; i< 4; i++){
  231: >                if (best_action_score[i] > max){
  232: >                        max = best_action_score[i];
  233: >                        maxIndex = i;
  234: >                }
  235: >
  236: >                if (((best_action_score[i] - max)>=-EPSINON) && ((best_action_score[i] - max) <= EPSINON
) ) {
  237: >                        tie = i;
  238: >                }
  239: >
  240: >                if (best_action_score[maxIndex] > best_action_score[tie]){
  241: >                        tie = maxIndex;
  242: >                }
  243: >
  244: >                //beak the tie randomly
  245: >                int k = rand() % 2;
  246: >                if (k == 0){
  247: >                        best_action = tie;
  248: >                }
  249: >                if (k == 1){
  250: >                        best_action = maxIndex;
  251: >                }
  252: >
  253: >        }
  254: >
  255: >
```

```
  256: >       end = clock();
  257: >     cost = (double)(end - begin)/CLOCKS_PER_SEC;
  258: >
  259: >       // output to file
  260: >       FILE *fp;
  261: >       fp=fopen("out.txt","w");
  262: >
  263: >       if(fp==NULL) {
  264: >               printf("File cannot open! " );
  265: >               exit(0);
  266: >       }
  267: >       fprintf(fp,"MaxDepth: %d\n", max_depth);
  268: >       fprintf(fp,"TotalExpanded: %d\n", expanded_nodes);
  269: >       fprintf(fp,"Generated nodes: %d\n", generated_nodes);
  270: >       fprintf(fp,"Budget: %d\n", budget);
  271: >       fprintf(fp,"Generated nodes: %d\n", generated_nodes);
  272: >       fprintf(fp,"time cost is: %lf secs", cost);
  273: >       fclose(fp);
  274: 147a355,356
  275: >
  276: >
  277: Only in ./src: .DS_Store
  278: diff -arw -x '*.o' -x '*.txt' /home/jovyan/base/old/notflat/src/pacman.c ./src/pacman.c
  279: 23d22
  280: < #include <assert.h>
  281: 30d28
  282: <
  283: 67d64
  284: < SCREEN *mainScreen = NULL;
  285: 70c67
  286: < enum { Wall = 1, Normal = 2, Pellet = 3, PowerUp = 4, GhostWall = 5, Ghost1 = 6, Ghost2 = 7, Ghost3 =
8, Ghost4 = 9, BlueGhost = 10, Pacman = 11 };
  287: ---
  288: > enum { Wall = 1, Normal, Pellet, PowerUp, GhostWall, Ghost1, Ghost2, Ghost3, Ghost4, BlueGhost, Pacman
 };
  289: 82a80
  290: >
  291: 103a102,103
  292: >     clock_t begin, end;
  293: >     double cost;
  294: 105a106,107
  295: >
  296: >
  297: 169a172
  298: >             begin = clock();
  299: 173a177,179
  300: >     end = clock();
  301: >     cost = (double)(end - begin)/CLOCKS_PER_SEC;
  302: >
  303: 265d270
  304: <                 delscreen(mainScreen);
  305: 316,317c321
  306: <     //char chr = ' ';                 //Variable used to display certain characters
  307: <     chtype chr = 'x';
  308: ---
  309: >     char chr = ' ';                 //Variable used to display certain characters
  310: 321d324
  311: <     wrefresh(win);
  312: 334,340d336
  313: <         //assert(mvwaddch(win, a, b, chr | attr) != ERR);
  314: <         // Don't write over where the other items will be (avoids a bit of flashing)
  315: <         if ((a != Loc[0][0] || b != Loc[0][1]) &&
  316: <             (a != Loc[1][0] || b != Loc[1][1]) &&
  317: <             (a != Loc[2][0] || b != Loc[2][1]) &&
  318: <             (a != Loc[3][0] || b != Loc[3][1]) &&
  319: <             (a != Loc[4][0] || b != Loc[4][1])){
  320: 342,343d337
  321: <             wrefresh(win);
  322: <         }
  323: 370d363
  324: <         wrefresh(win);
  325: 372d364
  326: <         wrefresh(win);
  327: 374d365
  328: <         wrefresh(win);
  329: 376d366
  330: <         wrefresh(win);
  331: 383d372
  332: <         wrefresh(win);
  333: 385d373
  334: <         wrefresh(win);
  335: 387d374
  336: <         wrefresh(win);
  337: 389d375
  338: <         wrefresh(win);
  339: 393,394c379
```

```
340: <     wattron(win, COLOR_PAIR(Pacman)); mvwaddch(win, Loc[4][0], Loc[4][1], 'C' | A_NORMAL);
341: <
342: ---
343: >     wattron(win, COLOR_PAIR(Pacman)); mvwaddch(win, Loc[4][0], Loc[4][1], 'C');
344: 410c395
345: <     chtype chr = 'x';                 //Variable used to display certain characters
346: ---
347: >     char chr = ' ';                 //Variable used to display certain characters
348: 426,430d410
349: <         if ((a != state.Loc[0][0] || b != state.Loc[0][1]) &&
350: <             (a != state.Loc[1][0] || b != state.Loc[1][1]) &&
351: <             (a != state.Loc[2][0] || b != state.Loc[2][1]) &&
352: <             (a != state.Loc[3][0] || b != state.Loc[3][1]) &&
353: <             (a != state.Loc[4][0] || b != state.Loc[4][1])){
354: 432,433d411
355: <             wrefresh(win);
356: <         }
357: 458d435
358: <         wrefresh(win);
359: 460d436
360: <         wrefresh(win);
361: 462d437
362: <         wrefresh(win);
363: 464d438
364: <         wrefresh(win);
365: 471d444
366: <         wrefresh(win);
367: 473d445
368: <         wrefresh(win);
369: 475d446
370: <         wrefresh(win);
371: 477d447
372: <         wrefresh(win);
373: 481,483c451
374: <     wattron(win, COLOR_PAIR(Pacman));
375: <     chr = 'C';
376: <     mvwaddch(win, state.Loc[4][0], state.Loc[4][1], chr | A_NORMAL);
377: ---
378: >     wattron(win, COLOR_PAIR(Pacman)); mvwaddch(win, state.Loc[4][0], state.Loc[4][1], 'C');
379: 496d463
380: <     delscreen(mainScreen);
381: 498a466
382: >     printf("time%f\n", cost);
383: 625,627c593
384: <     //initscr();                      //Needed for ncurses windows
385: <     mainScreen = newterm(getenv("TERM"), stdout, stdin);
386: <     set_term(mainScreen);
387: ---
388: >     initscr();                      //Needed for ncurses windows
389: diff -arw -x '*.o' -x '*.txt' /home/jovyan/base/old/notflat/src/utils.h ./src/utils.h
390: 19a20,21
391: >  * Variables have the same name as the global variables in pacman.c
392: >  * representing the state of the game.
393: 33d34
394: <
395: [2019/10/27 13:33:03] Attempting to make.
396: gcc  -g  -std=gnu99  -c -o src/utils.o src/utils.c
397: gcc  -g  -std=gnu99  -c -o src/priority_queue.o src/priority_queue.c
398: gcc  -g  -std=gnu99  -c -o src/ai.o src/ai.c
399: gcc  -g  -std=gnu99  -c -o src/pacman.o src/pacman.c
400: gcc -o pacman src/utils.o src/priority_queue.o src/ai.o src/pacman.o  -g  -std=gnu99 -lncurses -lm
401: [2019/10/27 13:33:03] Beginning tests.
402: rm: cannot remove '/home/scratch/_regOut.txt': No such file or directory
403: rm: cannot remove '/home/scratch/_valgrind.txt': No such file or directory
404: [2019/10/27 13:33:03] timeout 240 valgrind ./pacman /home/jovyan/a2/abyss100 ai max 100
405: [2019/10/27 13:33:04] Applied window width checking ignore patch.
406: 267c267
407: <         int h, w; getmaxyx(stdscr, h, w);
408: ---
409: >         int h = 35, w = 29; /*  getmaxyx(stdscr, h, w); */
410: [2019/10/27 13:33:04] Rebuild pacman.
411: gcc  -g  -std=gnu99  -c -o src/utils.o src/utils.c
412: gcc  -g  -std=gnu99  -c -o src/priority_queue.o src/priority_queue.c
413: gcc  -g  -std=gnu99  -c -o src/ai.o src/ai.c
414: gcc  -g  -std=gnu99  -c -o src/pacman.o src/pacman.c
415: gcc -o pacman src/utils.o src/priority_queue.o src/ai.o src/pacman.o  -g  -std=gnu99 -lncurses -lm
416: [2019/10/27 13:33:04] timeout 240 valgrind ./pacman /home/jovyan/a2/abyss100 ai max 100
417: [2019/10/27 13:33:35] Passed test!
418: [2019/10/27 13:33:35] timeout 240 valgrind ./pacman /home/jovyan/a2/abyss100 ai avg 100
419: [2019/10/27 13:37:37] Timed out test "abyss100", trying 2 more times.
420: [2019/10/27 13:37:37] timeout 240 valgrind ./pacman /home/jovyan/a2/abyss100 ai avg 100
421: [2019/10/27 13:41:40] Timed out test "abyss100", trying 1 more times.
422: [2019/10/27 13:41:40] timeout 240 valgrind ./pacman /home/jovyan/a2/abyss100 ai avg 100
423: [2019/10/27 13:45:43] Test "abyss100" timed out, no more retries left.
424: [2019/10/27 13:45:43] Attempting test "abyss100" without valgrind.
425: [2019/10/27 13:45:43] timeout 240 ./pacman /home/jovyan/a2/abyss100 ai avg 100
```

```
426: [2019/10/27 13:49:43] Test "abyss100" timed out even without valgrind.
427: [2019/10/27 13:49:43] timeout 240 valgrind ./pacman /home/jovyan/a2/column100 ai max 100
428: [2019/10/27 13:49:48] Passed test!
429: [2019/10/27 13:49:48] timeout 240 valgrind ./pacman /home/jovyan/a2/column100 ai avg 100
430: [2019/10/27 13:49:53] Passed test!
431: [2019/10/27 13:49:53] timeout 240 valgrind ./pacman /home/jovyan/a2/columnTrail ai max 100
432: [2019/10/27 13:50:01] Passed test!
433: [2019/10/27 13:50:01] timeout 240 valgrind ./pacman /home/jovyan/a2/columnTrail ai avg 100
434: [2019/10/27 13:50:11] Passed test!
435: [2019/10/27 13:50:11] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai avg 5
436: [2019/10/27 13:50:19] Passed test!
437: [2019/10/27 13:50:19] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai max 5
438: [2019/10/27 13:50:26] Passed test!
439: [2019/10/27 13:50:26] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai avg 1000
440: [2019/10/27 13:54:27] Timed out test "line", trying 2 more times.
441: [2019/10/27 13:54:27] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai avg 1000
442: [2019/10/27 13:58:27] Timed out test "line", trying 1 more times.
443: [2019/10/27 13:58:27] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai avg 1000
444: [2019/10/27 14:02:28] Test "line" timed out, no more retries left.
445: [2019/10/27 14:02:28] Attempting test "line" without valgrind.
446: [2019/10/27 14:02:28] timeout 240 ./pacman /home/jovyan/a2/line ai avg 1000
447: [2019/10/27 14:02:35] Passed test!
448: [2019/10/27 14:02:35] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai max 1000
449: [2019/10/27 14:06:36] Timed out test "line", trying 2 more times.
450: [2019/10/27 14:06:36] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai max 1000
451: [2019/10/27 14:10:36] Timed out test "line", trying 1 more times.
452: [2019/10/27 14:10:36] timeout 240 valgrind ./pacman /home/jovyan/a2/line ai max 1000
453: [2019/10/27 14:14:42] Test "line" timed out, no more retries left.
454: [2019/10/27 14:14:42] Attempting test "line" without valgrind.
455: [2019/10/27 14:14:42] timeout 240 ./pacman /home/jovyan/a2/line ai max 1000
456: [2019/10/27 14:14:50] Passed test!
457: [2019/10/27 14:14:50] timeout 240 valgrind ./pacman /home/jovyan/a2/snake100 ai avg 100
458: [2019/10/27 14:18:54] Timed out test "snake100", trying 2 more times.
459: [2019/10/27 14:18:54] timeout 240 valgrind ./pacman /home/jovyan/a2/snake100 ai avg 100
460: [2019/10/27 14:22:58] Timed out test "snake100", trying 1 more times.
461: [2019/10/27 14:22:58] timeout 240 valgrind ./pacman /home/jovyan/a2/snake100 ai avg 100
462: [2019/10/27 14:27:02] Test "snake100" timed out, no more retries left.
463: [2019/10/27 14:27:02] Attempting test "snake100" without valgrind.
464: [2019/10/27 14:27:02] timeout 240 ./pacman /home/jovyan/a2/snake100 ai avg 100
465: [2019/10/27 14:31:02] Test "snake100" timed out even without valgrind.
466: [2019/10/27 14:31:02] timeout 240 valgrind ./pacman /home/jovyan/a2/snake100 ai max 100
467: [2019/10/27 14:31:18] Passed test!
468: [2019/10/27 14:31:18] timeout 240 valgrind ./pacman /home/jovyan/a2/snakeConfine ai max 5
469: [2019/10/27 14:31:50] Passed test!
470: [2019/10/27 14:31:50] timeout 240 valgrind ./pacman /home/jovyan/a2/snakeConfine ai avg 5
471: [2019/10/27 14:32:23] Passed test!
472: [2019/10/27 14:32:23] timeout 240 valgrind ./pacman /home/jovyan/a2/basicDown ai max 5
473: [2019/10/27 14:32:26] Passed test!
474: [2019/10/27 14:32:26] timeout 240 valgrind ./pacman /home/jovyan/a2/basicDown ai avg 5
475: [2019/10/27 14:32:29] Passed test!
476: [2019/10/27 14:32:29] timeout 240 valgrind ./pacman /home/jovyan/a2/basicUp ai max 5
477: [2019/10/27 14:32:33] Passed test!
478: [2019/10/27 14:32:33] timeout 240 valgrind ./pacman /home/jovyan/a2/basicUp ai avg 5
479: [2019/10/27 14:32:36] Passed test!
480: [2019/10/27 14:32:36] timeout 240 valgrind ./pacman /home/jovyan/a2/basicLeft ai max 5
481: [2019/10/27 14:32:40] Passed test!
482: [2019/10/27 14:32:40] timeout 240 valgrind ./pacman /home/jovyan/a2/basicLeft ai avg 5
483: [2019/10/27 14:32:43] Passed test!
484: [2019/10/27 14:32:43] timeout 240 valgrind ./pacman /home/jovyan/a2/basicRightTwo ai max 5
485: [2019/10/27 14:32:47] Passed test!
486: [2019/10/27 14:32:47] timeout 240 valgrind ./pacman /home/jovyan/a2/basicRightTwo ai avg 5
487: [2019/10/27 14:32:50] Passed test!
488: [2019/10/27 14:32:50] Completed testing, tests succeeded: 20 / 22
```