

Fundamental Exercise on Computer and Information Engineering 1B Ncurses Game

XL15613 Thiago Machado da Silva

July 22, 2015

Output

In Figure 1 the game is shown, where a Game Over occurs. When I enlarge the screen this big, the game becomes too hard to win.

Source codes

Shown in Figure 2.



Figure 1: Game execution.

```

1 #include <ncurses.h>
2 #include <time.h> // for the time().
3 #include <stdlib.h>
4 #include <unistd.h> // for the sleep.
5
6 struct position {
7     int x, y;
8 };
9
10 typedef struct position POSITION;
11
12 // generalized "positioned" print.
13 void printPosition(POSITION *p, char *f) {
14     mvprintw(p->y, p->x, f);
15 }
16
17 // generalized "unit" initialization.
18 void initPositionLocation(POSITION *p,
19     int x, int y, char *f) {
20     printPosition(p, " ");
21     p->x = x;
22     p->y = y;
23     printPosition(p, f);
24 }
25
26 // generalized "inside the screen" movement.
27 void getPositionLocation(POSITION *c, int dx, int dy) {
28     int x2 = c->x + dx, y2 = c->y + dy;
29     c->x += dx;
30     (x2 < 0 ? -x2 : x2 > COLS - 1 ? COLS - x2 - 1 : 0);
31     c->y += dy;
32     (y2 < 0 ? -y2 : y2 > LINES - 1 ? LINES - y2 - 1 : 0);
33 }
34
35 // return true if the position of a and b are equal.
36 bool samePosition(POSITION *a, POSITION *b) {
37     return a->x == b->x && a->y == b->y;
38 }
39
40 // copy position from a to b.
41 void copyPosition(POSITION *a, POSITION *b) {
42     b->x = a->x;
43     b->y = a->y;
44 }
45
46 void initPlayerLocation(POSITION *player) {
47     initPositionLocation(player, 5, 5, "@");
48 }
49
50 void initMonsterLocation(POSITION *monster) {
51     initPositionLocation(monster,
52         COLS / 2, LINES / 2, "M");
53 }
54
55 void initTreasureLocation(POSITION *treasure,
56     int *numTreasure) {
57     *numTreasure = 0;
58     printPosition(treasure, " ");
59 }
60
61 void getPlayerLocation(POSITION *player, int key) {
62     getPosLocation(player,
63         key == KEY_LEFT ? -1 : key == KEY_RIGHT ? 1 : 0,
64         key == KEY_UP ? -1 : key == KEY_DOWN ? 1 : 0);
65 }
66
67 void getMonsterLocation(POSITION *monster,
68     POSITION *treasure, int *numTreasure) {
69     POSITION origin;
70     copyPosition(monster, &origin);
71     getPosLocation(monster,
72         rand() % 3 - 1, rand() % 3 - 1);
73
74     // don't let the monster walk to the treasure
75     if (*numTreasure == 1 &&
76         samePosition(monster, treasure)) {
77         copyPosition(&origin, monster);
78     }
79
80     // create a treasure
81     int dice = rand() % 10;
82     if (dice == 0 && *numTreasure == 0 &&
83         !samePosition(monster, &origin)) {
84         *numTreasure = 1;
85         copyPosition(&origin, treasure);
86         printPosition(treasure, "T");
87     }
88 }
89
90 int getTreasure(POSITION *player,
91     POSITION *treasure, int *numTreasure) {
92     int same = samePosition(player, treasure);
93     if (*numTreasure && same) {
94         // erase treasure from screen.
95         printPosition(treasure, " ");
96         *numTreasure--;
97     }
98     return same;
99 }
100
101 void encounter(POSITION *player, POSITION *monster) {
102     return samePosition(player, monster);
103 }
104
105 char gameOver() {
106     POSITION c = {COLS / 2 - 4, LINES / 2 - 4};
107     printPosition(&c, "GAME OVER");
108     refresh();
109     sleep(2);
110     return 'q';
111 }
112
113 #include <ncurses.h>
114 #include <time.h> // for the time()
115 #include <stdlib.h>
116 #include <math.h>
117 #include "redraw2.h"
118
119 int caughtByZombie(POSITION *player, POSITION *zombie,
120     int zombieNum) {
121     // for each zombie..
122     for(int i = 0; i < zombieNum; i++) {
123         if (samePosition(player, &zombie[i])) {
124             // some zombie encountered with the player.
125             return 1;
126         }
127     }
128     return 0; // no encounters.
129 }
130
131 void initZombie(POSITION *zombie, int zombieNum) {
132     for(int i = 0; i < zombieNum; i++) {
133         initPositionLocation(&zombie[i],
134             rand() % 20 - 10 + COLS / 2,
135             rand() % 20 - 10 + LINES / 2, "Z");
136     }
137 }
138
139 // distances to be walked, from b towards a.
140 POSITION distance(POSITION *a, POSITION *b,
141     int speed) {
142     // difference in position.
143     POSITION d = {a->x - b->x, a->y - b->y};
144     // hypotenuse.
145     double sq = sqrt(d.x * d.x + d.y * d.y);
146     d.x = speed * d.x / sq;
147     d.y = speed * d.y / sq;
148     return d;
149 }
150
151 void getZombieLocation(POSITION *zombie, int i,
152     POSITION *player, POSITION *treasure,
153     int *numTreasure) {
154     POSITION d = distance(player, &zombie[i],
155         rand() % 2 + 1);
156     getPositionLocation(&zombie[i], d.x, d.y);
157
158     // erase encountered treasures by zombies.
159     if (samePosition(&zombie[i], treasure)) {
160         *numTreasure = 0;
161     }
162 }
163
164 int getHighScore(int highScore, int score) {
165     return score > highScore ? score : highScore;
166 }
167
168 void loadHighScore() {
169     FILE *fp;
170     fp = fopen("highscore.txt", "r");
171     if (fp == NULL) {
172         return 0;
173     }
174     int score = 0;
175     fscanf(fp, "%d", &score);
176     fclose(fp);
177     return score;
178 }
179
180 void saveHighScore(int score) {
181     FILE *fpw;
182     fpw = fopen("highscore.txt", "w");
183     if (fpw == NULL) {
184         return;
185     }
186     fprintf(fpw, "%d", score);
187     fclose(fpw);
188 }
189
190 void highScoreDisp(int highScore) {
191     mvprintw(1, COLS / 2 + 2, "HI SCORE : %d",
192         highScore);
193 }
194
195 #include <ncurses.h>
196 #include "zombie.h"
197 #include "redraw2.h"
198
199 int main(int argc, char** argv) {
200     // 25 lines: declaration of variables
201     int zombieNum = 5;
202     initZombie(zombie, zombieNum);
203     // 4 lines: First, let's read key.
204     while (ch != 'q') {
205         mvprintw(1, 2, "stage : %d", (stage+1));
206         highScoreDisp(score);
207         mvprintw(3, 2, "count : %3d", MAX_COUNT - cnt);
208         // 6 lines: cnt++;
209         // move zombies
210         for (int i = 0; i < zombieNum; i++) {
211             getPositionLocation(&zombie[i],
212                 treasure, &numTreasure);
213             printPosition(&zombie[i], "Z");
214         }
215         // move player
216         printPosition(player->y, player->x, " ");
217         getPlayerLocation(player, ch);
218         mvprintw(player->y, player->x, "@");
219         refresh();
220     }
221     // 8 lines: judgement
222     if (caughtByZombie(player, zombie,
223         zombieNum)) {
224         // caught by zombie
225         ch = gameOver();
226         // 8 lines: else if (clear==stage+1)
227         saveHighScore(getHighScore(loadHighScore(),
228             score));
229     }
230     // 12 lines: cnt=0;
231 }

```

Figure 2: from left to right, top to bottom: redraw2.c (until line 65), redraw2.c (from line 66), zombie.c (until line 65), zombie.c (from line 66) and ncurses-game3.c (only the highlights).