

```

1: #include "defs.h"
2: #include <string.h>
3: #include "game.h"
4: #include "LCDG.h"
5: #include "switch.h"
6: #include "Timer.h"
7: #include "Music.h"
8: #include "xbee.h"
9:
10: #define VERTICAL 0
11: #define HORIZONTAL 1
12:
13: typedef struct {
14:     unsigned char x;
15:     unsigned char y;
16:     unsigned char orientation;
17:     unsigned char size;
18:     unsigned char hits;
19: } ShipType;
20:
21: typedef struct {
22:     unsigned int type:1;
23:     unsigned int x:4;
24:     unsigned int y:4;
25: } AttackType;
26:
27: CursorType cursor;
28:
29: static int state;
30:
31: static int buttonFlag;
32:
33: static char string[10];
34:
35: static ShipType ships[5] = {
36:     {0, 0, VERTICAL, 2, 0},
37:     {0, 0, VERTICAL, 3, 0},
38:     {0, 0, VERTICAL, 3, 0},
39:     {0, 0, VERTICAL, 4, 0},
40:     {0, 0, VERTICAL, 5, 0}
41: };
42:
43: static ShipType computerShips[5] = {
44:     {0, 0, VERTICAL, 2, 0},
45:     {0, 0, VERTICAL, 3, 0},
46:     {0, 0, VERTICAL, 3, 0},
47:     {0, 0, VERTICAL, 4, 0},
48:     {0, 0, VERTICAL, 5, 0}
49: };
50:
51: static unsigned char field[10][10];
52:
53: static int numShips;
54:
55: static AttackType enemyAttacks[100];
56: static int numEnemyAttacks;
57:
58: static AttackType playerAttacks[100];
59: static int numPlayerAttacks;
60:
61: int findValidPos(ShipType * array, int index);
62:
63: int checkDead(ShipType * array) {
64:     int i;
65:     for(i=0; i<5; i++) {
66:         if(array[i].size != array[i].hits) {
67:             return 0;
68:         }
69:     }
70:     return 1;
71: }
72:
73: void incState(void) {
74:     switch(state) {
75:         case WELCOME:
76:             state = WAITING_FOR_OPPONENT;
77:             break;
78:         case WAITING_FOR_OPPONENT:

```

```

79:         numShips = 1;
80:         state = PLACING_SHIPS;
81:         LCD_Clear(0);
82:         break;
83:     case PLACING_SHIPS:
84:         cursor.x = 0;
85:         cursor.y = 0;
86:         state = PLAYER_TURN_WAITING;
87:         LCD_Clear(0);
88:         strcpy(string, " ATTACK!!!");
89:         break;
90:     case PLAYER_TURN_DONE:
91:         if(checkDead(computerShips)) {
92:             state = WIN;
93:         }
94:         else {
95:             state = OPPONENT_TURN_WAITING;
96:             strcpy(string, " Opponent");
97:         }
98:         break;
99:     case COMPUTER_SCREEN:
100:         state = PLAYER_TURN_WAITING;
101:         break;
102:     case OPPONENT_TURN_WAITING:
103:         state = OPPONENT_TURN_DONE;
104:         break;
105:     case OPPONENT_TURN_DONE:
106:         if(checkDead(ships)) {
107:             state = LOSE;
108:         }
109:         else {
110:             state = PLAYER_TURN_WAITING;
111:             strcpy(string, " ATTACK!!!");
112:         }
113:         break;
114:     }
115:     Game_Update();
116: }
117:
118: unsigned char random(unsigned char max) {
119:     unsigned static char seed1 = 0;
120:     unsigned static char seed2;
121:     unsigned static short last = 0;
122:
123:     unsigned short tcnt = TCNT;
124:     seed1 = (tcnt&0xFF00) >> 8;
125:     seed2 = (tcnt&0x00FF);
126:
127:     last = ((unsigned short) seed1)*last + seed2;
128:
129:     return (unsigned char) (last%max);
130: }
131:
132: int shipInBounds(ShipType * array, int index) {
133:     ShipType * ship = &array[index];
134:
135:     if(ship->x < 0 || ship->x > 9 || ship->y < 0 || ship->y > 9 ||
136:        (ship->orientation == VERTICAL && ship->x + ship->size > 10) ||
137:        (ship->orientation == HORIZONTAL && ship->y + ship->size > 10)) {
138:         return 0;
139:     }
140:
141:     return 1;
142: }
143:
144: int checkHit(ShipType * array, int x, int y) {
145:     int i, j;
146:     for(i=0; i<5; i++) {
147:         for(j=0; j<array[i].size; j++) {
148:             if(array[i].orientation == HORIZONTAL) {
149:                 if(x == array[i].x && y == array[i].y + j) {
150:                     return i;
151:                 }
152:             }
153:             else if(x == array[i].x + j && y == array[i].y) {
154:                 return i;
155:             }
156:         }
157:     }

```

```

157:     }
158:
159:     return -1;
160: }
161:
162: int validShipPos(ShipType * array, int index) {
163:     ShipType ship = array[index];
164:     int i;
165:
166:     for(i=0; i<index; i++) {
167:         if(ship.orientation == HORIZONTAL) {
168:             if(array[i].orientation == HORIZONTAL) {
169:                 if(ship.x == array[i].x) {
170:                     if((ship.y + ship.size > array[i].y &&
171:                         ship.y + ship.size <= array[i].y + array[i].size) ||
172:                         (ship.y >= array[i].y &&
173:                         ship.y < array[i].y + array[i].size) ||
174:                         (array[i].y + array[i].size > ship.y &&
175:                         array[i].y + array[i].size <= ship.y + ship.size) ||
176:                         (array[i].y >= ship.y &&
177:                         array[i].y < ship.y + ship.size)) {
178:
179:                         return 0;
180:                     }
181:                 }
182:             }
183:         }
184:         else {
185:             if(ship.x >= array[i].x &&
186:                 ship.x < array[i].x + array[i].size &&
187:                 array[i].y >= ship.y &&
188:                 array[i].y < ship.y + ship.size) {
189:
190:                 return 0;
191:             }
192:         }
193:     }
194:     else {
195:         if(array[i].orientation == HORIZONTAL) {
196:             if(ship.y >= array[i].y &&
197:                 ship.y < array[i].y + array[i].size &&
198:                 array[i].x >= ship.x &&
199:                 array[i].x < ship.x + ship.size) {
200:
201:                 return 0;
202:             }
203:         }
204:         else {
205:             if(ship.y == array[i].y) {
206:                 if((ship.x + ship.size > array[i].x &&
207:                     ship.x + ship.size <= array[i].x + array[i].size) ||
208:                     (ship.x >= array[i].x &&
209:                     ship.x < array[i].x + array[i].size) ||
210:                     (array[i].x + array[i].size > ship.x &&
211:                     array[i].x + array[i].size <= ship.x + ship.size) ||
212:                     (array[i].x >= ship.x &&
213:                     array[i].x < ship.x + ship.size)) {
214:
215:                     return 0;
216:                 }
217:             }
218:         }
219:     }
220: }
221: return 1;
222: }
223:
224: void createField(ShipType * shipArray, int shipSize, AttackType * attackArray, int attackSize)
225: {
226:     int i, j;
227:
228:     for(i=0; i<10; i++) {
229:         for(j=0; j<10; j++) {
230:             field[i][j] = EMPTY;
231:         }
232:     }
233:     for(i=0; i<shipSize; i++) {

```

```

234:     ShipType ship = shipArray[i];
235:     if(ship.orientation == HORIZONTAL) {
236:         field[ship.x][ship.y] = SHIPEND_LEFT;
237:         for(j=1; j<ship.size-1; j++) {
238:             field[ship.x][ship.y+j] = SHIP_HORIZ;
239:         }
240:         field[ship.x][ship.y+ship.size-1] = SHIPEND_RIGHT;
241:     }
242:     else {
243:         field[ship.x][ship.y] = SHIPEND_UP;
244:         for(j=1; j<ship.size-1; j++) {
245:             field[ship.x+j][ship.y] = SHIP_VERT;
246:         }
247:         field[ship.x+ship.size-1][ship.y] = SHIPEND_DOWN;
248:     }
249: }
250:
251: for(i=0; i<attackSize; i++) {
252:     AttackType attack = attackArray[i];
253:     field[attack.x][attack.y] = attack.type;
254: }
255: }
256:
257: void enemyInit(void) {
258:     int i;
259:
260:     for (i=0; i<5; i++) {
261:         ShipType * ship = &computerShips[i];
262:         ship->x = random(10);
263:         ship->y = random(10);
264:         ship->orientation = random(2);
265:
266:         findValidPos(computerShips, i);
267:     }
268: }
269:
270: void enemyPickMove(void) {
271:     int i, x, y, moveFlag, hit;
272:
273:     do {
274:         moveFlag = 0;
275:         x = random(10);
276:         y = random(10);
277:
278:         for(i=0; i<numEnemyAttacks; i++) {
279:             if(enemyAttacks[i].x == x && enemyAttacks[i].y == y) {
280:                 moveFlag = 1;
281:             }
282:         }
283:     }while(moveFlag);
284:
285:     enemyAttacks[numEnemyAttacks].x = x;
286:     enemyAttacks[numEnemyAttacks].y = y;
287:     hit = checkHit(ships, x, y);
288:     if(hit == -1) {
289:         enemyAttacks[numEnemyAttacks++].type = MISS;
290:         strcpy(string, "    Miss ");
291:     }
292:     else {
293:         enemyAttacks[numEnemyAttacks++].type = HIT;
294:         strcpy(string, "    Hit ");
295:         Music_EnableOC7(EXPLODE);
296:         asm cli
297:     }
298: }
299:
300: void Game_Init(void) {
301:     state = WELCOME;
302:     numShips = 0;
303:     numEnemyAttacks = 0;
304:     numPlayerAttacks = 0;
305:     cursor.x = 0;
306:     cursor.y = 0;
307:     LED_DDR0 = 1;
308:     LED_DDR1 = 1;
309:     LED_DDR2 = 1;
310:     LED_DDR3 = 1;
311:     LED_DDR4 = 1;

```

```

312:     LED_DDR5 = 1;
313:
314:     LED0 = 1;
315:     LED1 = 1;
316:     LED2 = 1;
317:     LED3 = 1;
318:     LED4 = 1;
319:     LED5 = 1;
320:     Game_Update();
321: }
322:
323: void Game_Update(void) {
324:     int frameFlag = 1;
325:     switch(state) {
326:         case WELCOME:
327:             LCD_Clear(0);
328:             LCD_GoTo(4, 1);
329:             LCD_OutString("        Welcome        ");
330:             LCD_GoTo(5, 1);
331:             LCD_OutString("    to Battleship    ");
332:             Timer_Wait10ms(100);
333:             incState();
334:             break;
335:         case WAITING_FOR_OPPONENT:
336:             enemyInit();
337:             incState();
338:             break;
339:         case PLACING_SHIPS:
340:             //LCD_Clear(0);
341:             LCD_GoTo(3,1);
342:             LCD_OutString("    Place    ");
343:             LCD_GoTo(4,1);
344:             LCD_OutString("    your    ");
345:             LCD_GoTo(5,1);
346:             LCD_OutString("    ships    ");
347:             createField(ships, numShips, enemyAttacks, 0);
348:             LCD_DrawGrid(field);
349:             break;
350:         case PLAYER_TURN_WAITING:
351:             //LCD_Clear(0);
352:             LCD_GoTo(4,1);
353:             LCD_OutString(string);
354:             createField(ships, 0, playerAttacks, numPlayerAttacks);
355:             LCD_DrawGrid(field);
356:             break;
357:         case PLAYER_TURN_DONE:
358:             //LCD_Clear(0);
359:             LCD_GoTo(4,1);
360:             LCD_OutString(string);
361:             createField(ships, 0, playerAttacks, numPlayerAttacks);
362:             LCD_DrawGrid(field);
363:             break;
364:         case OPPONENT_TURN_WAITING:
365:             //LCD_Clear(0);
366:             createField(ships, numShips, enemyAttacks, numEnemyAttacks);
367:             LCD_GoTo(4,1);
368:             LCD_OutString(string);
369:             LCD_DrawGrid(field);
370:             Music_EnableOC7(WHISTLE);
371:             asm cli
372:             Timer_Wait10ms(102);
373:             enemyPickMove();
374:             incState();
375:             break;
376:         case OPPONENT_TURN_DONE:
377:             //LCD_Clear(0);
378:             createField(ships, numShips, enemyAttacks, numEnemyAttacks);
379:             LCD_GoTo(4,1);
380:             LCD_OutString(string);
381:             LCD_DrawGrid(field);
382:             Timer_Wait10ms(140);
383:             incState();
384:             break;
385:         case COMPUTER_SCREEN:
386:             //LCD_Clear(0);
387:             createField(computerShips, 5, playerAttacks, numPlayerAttacks);
388:             LCD_DrawGrid(field);
389:             break;

```

```

390:     case WIN:
391:         //LCD_Clear(0);
392:         LCD_GoTo(4, 1);
393:         LCD_OutString("          You Win          ");
394:         break;
395:     case LOSE:
396:         //LCD_Clear(0);
397:         LCD_GoTo(4, 1);
398:         LCD_OutString("          You Lose          ");
399:         break;
400:     }
401: }
402:
403: int findValidPos(ShipType * array, int index) {
404:     if(validShipPos(array, index) && shipInBounds(array, index)) {
405:         return 1;
406:     }
407:     else {
408:         ShipType * ship = &array[index];
409:         unsigned int tempX = (ship->x + 9)%10;
410:         unsigned int tempY = (ship->y + 9)%10;
411:         unsigned int tempDir = ship->orientation ^ 1;
412:
413:         for(ship->orientation = tempDir ^ 1; ship->orientation != tempDir; ship->orientation = (++
ship->orientation)%2) {
414:             for(ship->x = (tempX+1)%10; ship->x != tempX; ship->x = (++ship->x)%10) {
415:                 for(ship->y = (tempY+1)%10; ship->y != tempY; ship->y = (++ship->y)%10) {
416:                     if(validShipPos(array, index) && shipInBounds(array, index)) {
417:                         return 1;
418:                     }
419:                 }
420:             }
421:         }
422:
423:         ship->x = (tempX+1)&0x0F;
424:         ship->y = (tempY+1)&0x0F;
425:         ship->orientation = (tempDir+1)&0x01;
426:
427:         return 0;
428:     }
429: }
430:
431: void flag(void) {
432:     buttonFlag = 0;
433: }
434:
435: void Game_DPad(unsigned char direction) {
436:     unsigned int tempX, tempY;
437:     if(!buttonFlag) {
438:         switch(state) {
439:             case PLACING_SHIPS:
440:                 tempX = ships[numShips-1].x;
441:                 tempY = ships[numShips-1].y;
442:
443:                 do {
444:                     switch(direction) {
445:                         case UP:
446:                             ships[numShips-1].x--;
447:                             break;
448:                         case DOWN:
449:                             ships[numShips-1].x++;
450:                             break;
451:                         case LEFT:
452:                             ships[numShips-1].y--;
453:                             break;
454:                         case RIGHT:
455:                             ships[numShips-1].y++;
456:                             break;
457:                     }
458:                 }while(!validShipPos(ships, numShips-1) && shipInBounds(ships, numShips-1));
459:
460:                 if(validShipPos(ships, numShips-1) && shipInBounds(ships, numShips-1)) {
461:                     Game_Update();
462:                 }
463:             else {
464:                 ships[numShips-1].x = tempX&0x0F;
465:                 ships[numShips-1].y = tempY&0x0F;
466:             }

```

```

467:         break;
468:     case PLAYER_TURN_WAITING:
469:         switch(direction) {
470:             case UP:
471:                 cursor.x = (cursor.x+9)%10;
472:                 break;
473:             case DOWN:
474:                 cursor.x = (cursor.x+1)%10;
475:                 break;
476:             case LEFT:
477:                 cursor.y = (cursor.y+9)%10;
478:                 break;
479:             case RIGHT:
480:                 cursor.y = (cursor.y+1)%10;
481:                 break;
482:         }
483:         Game_Update();
484:         break;
485:     case COMPUTER_SCREEN:
486:         incState();
487:         break;
488: }
489:
490:     buttonFlag = 1;
491:     enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
492: }
493: }
494:
495: void LEDflash(void) {
496:     int i;
497:     for(i=0; i<10; i++) {
498:         LED0 ^= 1;
499:         LED1 ^= 1;
500:         LED2 ^= 1;
501:         LED3 ^= 1;
502:         LED4 ^= 1;
503:         LED5 ^= 1;
504:         Timer_Wait1ms(100);
505:     }
506: }
507:
508: void Game_A(void) {
509:     int i, attackFlag;
510:     if(!buttonFlag) {
511:         switch(state) {
512:             case PLACING_SHIPS:
513:                 if(findValidPos(ships, numShips)) {
514:                     numShips++;
515:                 }
516:
517:                 if(numShips == 6) {
518:                     numShips--;
519:                     incState();
520:                 }
521:                 else {
522:                     Game_Update();
523:                 }
524:                 break;
525:             case PLAYER_TURN_WAITING:
526:                 attackFlag = 0;
527:                 for(i=0; i<numPlayerAttacks; i++) {
528:                     if(playerAttacks[i].x == cursor.x && playerAttacks[i].y == cursor.y) {
529:                         attackFlag = 1;
530:                     }
531:                 }
532:                 if(!attackFlag) {
533:                     int hit = checkHit(computerShips, cursor.x, cursor.y);
534:                     playerAttacks[numPlayerAttacks].x = cursor.x;
535:                     playerAttacks[numPlayerAttacks].y = cursor.y;
536:                     if(hit == -1) {
537:                         playerAttacks[numPlayerAttacks++].type = MISS;
538:                         state = PLAYER_TURN_DONE;
539:                         Music_EnableOC7(WHISTLE);
540:                         asm cli
541:                         Timer_Wait10ms(102);
542:                         strcpy(string, "    MISS!    ");
543:                         Game_Update();
544:                         Timer_Wait10ms(100);

```

```

545:         }
546:     else {
547:         computerShips[hit].hits++;
548:         playerAttacks[numPlayerAttacks++].type = HIT;
549:         state = PLAYER_TURN_DONE;
550:         Music_EnableOC7(WHISTLE);
551:         asm cli
552:         Timer_Wait10ms(102);
553:         strcpy(string, "    HIT!    ");
554:         Game_Update();
555:         Music_EnableOC7(EXPLODE);
556:         asm cli
557:         LEDflash();
558:     }
559:     incState();
560: }
561: break;
562: case COMPUTER_SCREEN:
563:     incState();
564:     break;
565: }
566:
567: buttonFlag = 1;
568: enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
569: }
570: }
571:
572: void Game_B(void) {
573:     if(!buttonFlag) {
574:         switch(state) {
575:             case PLACING_SHIPS:
576:                 ships[numShips-1].orientation ^= 1;
577:                 if(validShipPos(ships, numShips-1) && shipInBounds(ships, numShips-1)) {
578:                     Game_Update();
579:                 }
580:             else {
581:                 ships[numShips-1].orientation ^= 1;
582:             }
583:             break;
584:             case PLAYER_TURN_WAITING:
585:                 state = COMPUTER_SCREEN;
586:                 Game_Update();
587:                 break;
588:             case COMPUTER_SCREEN:
589:                 incState();
590:                 break;
591:         }
592:     }
593:
594:     buttonFlag = 1;
595:     enableOC6(&flag, DEBOUNCE_DELAY, 8, 1);
596: }
597: }
598:
599: CursorType Game_GetCursor(void) {
600:     return cursor;
601: }
602:
603: int Game_GetState(void) {
604:     return state;
605: }

```