

```

1  #include <iostream>
2  #include <string>
3  #include <cstdlib>
4  #include <ctime>
5  #include <bitset>
6  #include <math.h>
7  using namespace std;
8
9  // Author: Glenn Storm
10 // Text adventure game Zombie Cruise
11 //
12 // Object: survive on a cruise ship with a collection of people faced with a
mysterious virus that has been released
13 //
14 // Locations: [8] Bridge, Fore Deck, Aft Deck, Ballroom, Lounge, Kitchen, Store
Room, Engine Room
15 // Items: [8] Flare Gun, Fire Extinguisher, Alcohol Bottle, Diving Knife, Spear
Gun, Wrench, Cleaver, Fuel Can
16 // People: [8] Captain, 1st Mate, Chef, Bartender, Mr. Rich, Mrs. Rich, Prof.
Smart, Ms. Sass
17 //
18 // (see Revisions below)
19 // Game data: Time to Release (0-7 turns), Is Released, Radio Used, S.O.S. Called,
Rescue Arrived, Game Over
20 // Time / Scoring data: Time To Rescue (since call for help, 0-31 turns), Score
(0-31)
21 // Player data: Current Location, Left Equip, Right Equip, Health
22 // Location data: Location I.D., Exit A, Exit B, Exit C
23 // Item data: Item I.D., Current Location, Is Equipped, Is Used, Damage
24 // Character data: People I.D., Current Location, Is Infected, Is Zombie, Target
Location
25 //
26 // Working Memory Budget : 32 bytes (256 bits)
27 // Game: 1 byte
28 // Time / Score: 1 byte
29 // Player 1 byte
30 // Locations: 8 bytes
31 // Items: 13 bytes (1 of each item, plus extra one of 5 items)
32 // Characters: 8 bytes
33 //
34 // -- REVISIONS --
35 //
36 // [oops! 0-7 uses three bits, not two]
37 // Game data : (ok)
38 // Time / Score data : (ok)
39 // Player data : only one equip item? (not ok)
40 //      [currentLocation(3)-equipItem(3)-healthRemaining(2)]
41 //      borrow 1 byte from items, use for 2 equip plus inventory? (ok)
42 //
[currentLocation(3)-equipItemA(3)-equipItemB(3)-inventoryHeld(3)-healthRemaining(2)-extra
?(2)]
43 //      use extra 2 bits for your appearance when 'look at me' or 'look at mirror'?
("spiffy", "stressed", "worn out", "like hell") (not ok)
44 //      (refactor...)
45 //      use 4 bit number to index items equipped (0-11), use value 15 to equal 'none
equipped' (ok)
46 //      use 3 bit number to track how many items carried at once (7 max) (ok)
47 //
[currentLocation(3)-healthRemaining(2)-equipItemA(4)-equipItemB(4)-inventoryCarried(3)]
48 // Location data : no location i.d. needed? only two exits per location? (not ok)
49 //      [exitA(3)-exitB(3)-...]
50 //      borrow 1 byte from items, use 1 bit for each of 8 rooms? (ok - must assemble
3 bits from 2 different bytes, get/set num)
51 //      [exitA(3)-exitB(3)-exitC(2+1 bit from aux byte)]
52 // Item data : no damage rating needed? (ok) 11 bytes, 1 of each plus extra 3 items
total (ok)

```

```

53 //      [itemID(3)-currentLocation(3)-isHeld(1)-isUsed(1)]
54 // Character data : no char i.d. needed? (ok)
55 //      [currentLocation(3)-targetLocation(3)-isInfected(1)-isZombie(1)]
56 //
57 // [wait. data (like location data) should only include that which cannot be
static: exits are static]
58 // Location data: no exit data needed? (ok)
59 //      [fireStatus(2), fireDuration(3), locationDamage(2), lightsOn(1)]
60 // (this means no splitting location data, and +1 item available = total of 12
items, 12 bytes of item data)
61 //
62 // Revised Working Memory Budget : 32 bytes (256 bits)
63 // Game: 1 byte
64 // Time / Score: 1 byte
65 // Player 2 bytes
66 // Locations: 8 bytes
67 // Items: 12 bytes (1 of each item, plus extra one of 4 items)
68 // Characters: 8 bytes
69 //
70 // [wait. no way to kill characters/zombies with no health. need wait during
infection plus zombie plus death]
71 // REFACTOR NEEDED
72 // [need target location? can use a single 'wait here' bit to slow zombies and use
other two bits for health?]
73 // Character data: nix target room (3 bits) as unnecessary, use single 'wait 1
turn' bit, use other two for health 0-3 (ok)
74 // [currentLocation(3)-waitHere(1)-health(2)-isInfected(1)-isZombie(1)]
75 //
76 // [wait. location timer needs 4 bits, not 3]
77 // [lose location damage bits. transfer one bit to fire timer (4), call other bit
'flicker' to effect light on] (ok)
78 // [lightsOn(1)-flickerLights(1)-fireState(2)-fireTimer(4)]
79 //
80 // [wait. initial location description > flickering lights]
81 // [lose light flicker bit and use it to indicate if location has been visited]
82 // [lightsOn(1)-visited(1)-fireState(2)-fireTimer(4)]
83 //
84 // -- END REVISIONS --
85 //
86 // Boat Map:
87 //
88 //      +-----+
89 //      |  0  |
90 // +-----+-----+-----+
91 // |  2  |  4  |  3  |  1  /
92 // \-----+-----+-----/
93 //  x\  7  |  5  |  6  /
94 //      +-----+-----+
95 //
96 // 0= Bridge, 1= Fore Deck, 2= Aft Deck, 3= Ballroom, 4= Lounge, 5= Kitchen, 6=
Store Room, 7= Engine Room
97 // Exits: 0=1-2-4, 1=0-3-6, 2=0-4-7, 3=1-4-5, 4=0-2-3, 5=3-6-7, 6=1-5-7, 7=2-5-6
98 //
99 // Player Commands:
100 // Quit, Help, Wait, Look, Look At [item/character], Take [item], Drop [item],
Inventory,
101 // Equip [item], Use [item/location feature], Attack [character] With [item],
102 // Move To [location], Talk To [character]
103 //
104 // Player can have up to 2 items at once, Player can move, Player can use items,
Items can be moved or used, Characters can move
105 // Characters can be infected, Infected characters turn zombie, Zombies can move,
Zombies can attack player or characters
106 // Characters infected previous turn 'rest' at location one turn, Infected
characters turn zombie next turn
107 // Characters not infected move from locations with zombies, characters stay in

```

```

locations with other characters or player
108 // Zombies target location and take another turn to move, Zombies target exits to
follow characters or player
109 // Attacking Zombies with items does damage, Attacking Characters with items kills
them (no points)
110 // Items Spear Gun, Flare Gun, Fuel Can, Alcohol Bottle are one use only, destroyed
after use, auto-dropped at location
111 //
112 // Locations have a light switch, and if player uses, lights toggle on or off,
unless they are damaged
113 // While lights are off, room descriptions are unavailable with 'Look' command,
only sounds and smells are provided
114 // Flickering lights provide a more brief and vague description of the room and its
contents (characters/zombies)
115 // Item descriptions are only randomly available (50% chance) while lights are off
or flickering
116 // Locations on fire always have light. Flare gun used provides light that turn
117 //
118 // Fuel can and alcohol bottle used makes location flammable, Flare gun used in
flammable location makes fire (while items exist)
119 // Location fire status (0-3) can be none, flammable, onFire, burnt. Fire damage
(0-7) occurs in location over time (0-15 turns)
120 // If fire in location has damaged more than 2, can spread to flammable adjacent
locations
121 // If damaged more than 4, can spread to non-flammable locations, but at 7, fire
goes out on its own
122 // If location damaged more than 2, lights flicker if on, if more than 4, lights
cannot turn on
123 // Characters move from locations with fire (will continually exit if entire ship
on fire)
124 // If Bridge is on fire, player cannot use radio to call S.O.S. without taking 4
damage and dying
125 // Fire can spread to flammable locations connected with exit (chance per turn)
126 // Using Fire Extinguisher in location removes items that make fire
127 //
128 // Character health = 1
129 // Zombie health = 3
130 // Player health = 3
131 // If Player loses all health, - Player loses -
132 //
133 // Item Damage: Flare Gun=3, Fire Extinguisher=2, Alcohol Bottle=1, Diving Knife=2,
Spear Gun=3, Wrench=2, Cleaver=2, Fuel Can=1
134 // = ITEM DAMAGE TABLE =
135 //   Spear Gun, Flare Gun           : 3
136 //   Fire Extinguisher, Wrench,
137 //   Cleaver, Diving Knife         : 2
138 //   Alcohol Bottle, Fuel Can      : 1
139 // Fire Damage: 2 per turn stayed at location on fire (characters not infected will
always survive fire, by exiting)
140 //
141 // Player at Bridge can use radio (takes 2 turns)
142 // If Player uses radio at Bridge, S.O.S. called and timer started
143 // If Player survives time to rescue, coast guard arrives
144 // If <4 Zombies left, rescue successful, - Player wins -
145 // If >=4 Zombies left, coast guard overcome, - Player loses -
146 //
147 // Scoring:
148 // Character survives to successful rescue: 2 points
149 // Kill a Zombie: 3 points
150 // Player survives until rescue: 2 points
151 // Rescue successful (<4 Zombies at rescue): 5 points
152 // HIGHEST SCORE POSSIBLE : 24+2+5 = 31
153 // Score Ranks: <11 "Coward", 11-20 "Survivor", 21-30 "Hero", 31 "Zombie Killer"
154 //
155 // [REVISION SCORING 0-7, not 0-31]
156 // Player survives until rescue: 2 points

```

```

157 // Survivors saved (>3 passengers unturned): 2 points
158 // Rescue successful (<4 Zombies at rescue): 3 points
159 // HIGHEST SCORE POSSIBLE : 2+2+3 = 7
160 // Score Ranks: <3 "Zombie Meat", 3-4 "Survivor", 5-6 "Hero", 7 "Zombie Killer"
161 //
162 // Dialog:
163 // Characters say unique lines to player when talked to
164 // Characters say unique exchanges when together at location
165 // Infected Characters say subtle clue to excuse 'rest'
166 // Zombies say unique mumbled versions of character lines
167 // Player is silent
168 //
169 // Game Flow:
170 // 1. Introduction - Bon Voyage (8 turns)
171 // 2. Begin - Outbreak (food or drink?, which character 1st?)
172 // 3. Middle A - Havok (characters and items)
173 // 4. Plateau - S.O.S. (timer start)
174 // 5. Middle B - Survive (combat and hazards)
175 // 6. Crisis - Rescue? (timer end 32 turns)
176 // 7. End - Prologue and Scoring
177 //
178 // Dialog:
179 // Character dialog is presented in response to "Talk To" player commands or as a
result of two characters in the same location
180 // Attacked character exclaims as injury or dying words
181 // Infected character dialog is a subtle reference to needing to stop and rest
182 // Zombie dialog is a mumbled version of the character's normal dialog to player
183 // Dialog Format: [player response A] [player response B] [char to char]
[affirmative] [negative] [injury] [rest] [zombie speak]
184 // Captain: "Ahoy there" "If you need anything, just ask" "Is everyone all right?"
"Certainly" "I'm afraid not" "Oh!" "I need to rest" "Ahhuugh thuuugh"
185 // 1st Mate: "Here to help" "At your service" "Can I get anyone anything?" "Yes"
"Not really" "Ah!" "Let me sit down" "Huungh tuugh huuughp"
186 // Chef: "Stay out of my kitchen" "Bon appetit" "Is anyone hungry?" "Oui oui" "No"
"Sacre bleu!" "Excuse moi" "Buughn appuughtuuught"
187 // Bartender: "It's always happy hour" "I'm here to listen" "Another round?" "Yeah"
"Nope" "Hey!" "I'm going down" "Uuhts uuhlwuhs huuuhpugh huugh"
188 // Mr. Rich: "Bully!" "I thought this was a private cruise" "Do you know how much
money I have?" "Of course" "Not at all" "I say!" "I'm okay" "Buuulluughe"
189 // Mrs. Rich: "Well, I never!" "The service here is awful" "Can you help me?" "Yes
indeed" "Certainly not" "My stars!" "I'm feeling faint" ""
190 // Prof. Smart: "This is fascinating" "I have a theory" "Don't you see what this
means?" "I think so" "It's unknowable" "Stay back!" "I need a minute" "Thuughs ughs
fuuusuunughtugh"
191 // Ms. Sass: "Seriously?" "Worst cruise ever" "Could this be any more tragic?"
"Whatever" "Like I care" "Ew!" "Just leave me alone" "Surunghslughee"
192
193 class ZCPlayer {
194     public:
195         bitset<16> pBits;
196         // [ (3)currentLocation, (2)healthRemaining, (4)equipR, (4)equipL,
(3)inventoryStored ]
197 };
198
199 class ZCLocation {
200     public:
201         bitset<8> locBits;
202         // [ (1)lightsOn, (1)visited, (2)fireStatus, (4)fireDuration ]
203 };
204
205 class ZCChar {
206     public:
207         bitset<8> charBits;
208         // [ (3)currentLocation, (1)waitHere, (2)healthRemaining, (1)isInfected,
(1)isZombie ]
209 };

```

```

210
211 class ZCItem {
212     public:
213         bitset<8> itemBits;
214         // [ (3)itemID, (3)currentLocation, (1)isHeld, (1)isUsed ]
215 };
216
217 class ZCGame {
218     public:
219         bitset<8> gameBits; // 1 byte
220         bitset<8> scoreBits; // 1 byte
221         ZCPlayer player; // 2 bytes
222         ZCLocation* locations; // 1 bytes
223         ZCChar* characters; // 8 bytes
224         ZCItem* items; // 12 bytes
225         // 32 bytes total working memory
226
227         void Initialize();
228         int DisambiguateLocation( string* words );
229         int DisambiguateItem( string* words );
230         int DisambiguateCharacter( string* words );
231         int FindItemInLocation( int itemType );
232         int FindItemInInventory( int itemType );
233         void ParseMove( string pMove );
234         void IncrementTurn();
235         void IncrementScore( int scoreAdd );
236         bool IsDarkArea();
237         void LocationNotice();
238         void FireNotice();
239         void ItemNotice();
240         void CharacterNotice();
241         bool CharacterAlive( int charIndex );
242         void ZombieChatter();
243         void CharacterChatter();
244         string InventoryFormat();
245         bool TakeItem( int itemIndex );
246         bool DropItem( int itemIndex );
247         void EquipItem( int itemIndex );
248         void EquipAny();
249         bool UseItem( int itemIndex );
250         void InfectCharacter( int charIndex );
251         void IncrementStory();
252         void IncrementInfection();
253         void IncrementFire();
254         void FireDamage();
255         void DoZombieMoves();
256         void DoZombieAttacks();
257         void DoCharacterMoves();
258         void DoPlayerAttack();
259 };
260
261 bitset<8> SetNum( bitset<8> bits, int num, int startPos, int range ) {
262     bits &= ~( ( 1 << range ) - 1 ) << startPos ); // clear
263     bits |= ( num << startPos ); // set
264     return bits;
265 }
266
267 bitset<16> SetNum( bitset<16> bits, int num, int startPos, int range ) {
268     bits &= ~( ( 1 << range ) - 1 ) << startPos );
269     bits |= ( num << startPos );
270     return bits;
271 }
272
273 int GetNum( bitset<8> bits, int startPos, int range ) {
274     unsigned short tmp = bits.to_ulong(); // required pre-process
275     // static_cast<int>() required to convert double from pow()
276     return ( ( tmp >= startPos ) & ( static_cast<int>( pow( 2, range ) ) - 1 ) ); //

```

```

"(2^range)-1" = mask!
276 }
277
278 int      GetNum( bitset<16> bits, int startPos, int range ) {
279     unsigned short tmp = bits.to_ulong();
280     return ( ( tmp >>= startPos ) & ( static_cast<int>( pow( 2, range ) )-1 ) );
281 }
282
283 bitset<8>  SetBit( bitset<8> bits, bool b, bitset<8> mask ) {
284     return ( ( b ) ? ( bits |= mask ) : ( bits &= ~mask ) );
285 }
286
287 bitset<16> SetBit( bitset<16> bits, bool b, bitset<16> mask ) {
288     return ( ( b ) ? ( bits |= mask ) : ( bits &= ~mask ) );
289 }
290
291 bool      GetBit( bitset<8> bits, bitset<8> mask ) {
292     return ( ( bits & mask ) == mask );
293 }
294
295 bool      GetBit( bitset<16> bits, bitset<16> mask ) {
296     return ( ( bits & mask ) == mask );
297 }
298
299 string    DebugBits( string name, bitset<8> bits ) {
300     string retString = ". Debug Bits ";
301     retString += name + " [";
302     for ( int i=0; i<8; i++ ) {
303         if ( GetBit( bits, (1 << (7-i)) ) == 0 ) // reverse order
304             retString += "0";
305         else
306             retString += "1";
307     }
308     retString += "]\n";
309     return retString;
310 }
311
312 string    DebugBits( string name, bitset<16> bits ) {
313     string retString = ". Debug Bits ";
314     retString += name + " [";
315     for ( int i=0; i<16; i++ ) {
316         if ( GetBit( bits, (1 << (15-i)) ) == 0 )
317             retString += "0";
318         else
319             retString += "1";
320         if ( i == 7 )
321             retString += " ";
322     }
323     retString += "]\n";
324     return retString;
325 }
326
327 string InsertNewlines( string input, int charWidth ) {
328     string retString = input;
329     int pos = 0;
330     int nlPos = charWidth;
331     bool done = ( nlPos > retString.length() );
332     // start at first char position, step ahead charWidth, work backwards to find a
space and replace it with a newline, repeat
333     while ( !done ) {
334         for ( int i=(nlPos-1); i>pos; i-- ) {
335             if ( retString.substr( i, 1 ) == " " ) {
336                 // space found
337                 string pre = retString.substr( 0, i );
338                 string post = retString.substr( (i+1), retString.npos );
339                 // replaced with newline

```

```

340         retString = pre + "\n" + post;
341         // step forward charWidth from there
342         pos = (i+2);
343         nlPos = (pos+charWidth);
344         done = ( post.length() < charWidth );
345         break;
346     }
347 }
348 }
349 return retString;
350 }
351
352 string HelpFormat() {
353     string retString = "";
354
355     retString = "[HELP]";
356     retString += "\n\n";
357     retString += "Zombie Cruise is a text-based adventure, where player commands
take the form of\nkeywords.";
358     retString += "\n\n";
359     retString += "To win, you will have to consider what is happening in the game,
based on the\ntext presented. You will be able to perform actions and react to
your\nsurroundings if you read carefully.";
360     retString += "\n\n";
361     retString += "Here are the commands available to you:";
362     retString += "\n\n";
363     retString += " help";
364     retString += "\n";
365     retString += " wait";
366     retString += "\n";
367     retString += " look / around / at <location> / <item> / <character> / myself";
368     retString += "\n";
369     retString += " go/move/run/walk to <location>";
370     retString += "\n";
371     retString += " inventory";
372     retString += "\n";
373     retString += " take <item> / all";
374     retString += "\n";
375     retString += " drop <item> / all";
376     retString += "\n";
377     retString += " equip <item> / any (allows attack, also protects from attack)";
378     retString += "\n";
379     retString += " attack *NOTE: must have item equipped to attack";
380     retString += "\n";
381     retString += " use <item> / <location-specific feature>";
382     retString += "\n";
383     retString += " quit";
384     retString += "\n\n";
385     retString += "Good luck.";
386     retString += "\n";
387
388     return retString;
389 }
390
391 void ZCGame::Initialize() {
392     // gameBits
393     gameBits = 0 << 0;
394     // scoreBits
395     scoreBits = 0 << 0;
396     // playerBits
397     player.pBits = 0 << 0;
398     locations = new ZCLocation[8];
399     characters = new ZCChar[8];
400     items = new ZCItem[12];
401     for (int i=0; i<8; i++) {
402         // locations

```

```

403     locations[i].locBits = 1 << 7; // all lights on
404     // characters
405     characters[i].charBits = 1 << 5; // all start on foredeck
406     characters[i].charBits = SetBit( characters[i].charBits, true, (1 << 4) );
// all wait at start
407     characters[i].charBits = SetNum( characters[i].charBits, 3, 2, 2 ); // all
have 3 health remaining
408     // items
409     items[i].itemBits = i << 5; // one of each item
410     switch(i) {
411     case 0:
412         items[i].itemBits |= 0 << 2; // flare gun at bridge
413         break;
414     case 1:
415         items[i].itemBits |= 3 << 2; // fire extinguisher at ballroom
416         break;
417     case 2:
418         items[i].itemBits |= 4 << 2; // alcohol bottle at lounge
419         break;
420     case 3:
421         items[i].itemBits |= 1 << 2; // diving knife at fore deck
422         break;
423     case 4:
424         items[i].itemBits |= 2 << 2; // spear gun at aft deck
425         break;
426     case 5:
427         items[i].itemBits |= 7 << 2; // wrench at engine room
428         break;
429     case 6:
430         items[i].itemBits |= 5 << 2; // cleaver at kitchen
431         break;
432     case 7:
433         items[i].itemBits |= 6 << 2; // fuel can at store room
434         break;
435     }
436 }
437 for (int i=8; i<12; i++) {
438     items[i].itemBits = (rand() % 8) << 5; // random item
439     items[i].itemBits |= (rand() % 8) << 2; // random location at start
440 }
441 };
442
443 string StoryFormat( int storyStage ) {
444     string retString = "";
445     switch (storyStage) {
446     case 0:
447         retString = "It is a warm summer afternoon, and this evening's event is a
private moonlight cruise hosted by Doctor Zilch in honor of a breakthrough drug discovery
he plans to announce to selected guests and the press. You have been invited to cover the
story for the local newspaper. Here at the marina, the cruise ship 'Hot Irony' is ready
to recieve the small number of guests. A crew is aboard already to prepare the dinner
service for this evening. You board the ship, and make your way to the lounge with the
other guests. As the ship casts off, you note how beautiful the scene is as the setting
sun glistens on the ocean's horizon.\n";
448         break;
449     case 1:
450         retString = "The dinner bell is rung by Chef Rotisserie, and guests are
welcomed to a long table set up in the ballroom. The Captain takes the seat at the head
of the table, as a fine wine is poured for each guest by Dr. Zilch. 'This is a momentous
occassion, my friends. You are truly in for a treat! Heee hee heee!' Professor Smart
says, 'Well, we're all very excited to hear your announcement doctor.' 'Yes, yes. But
first, a toast! To all our lessons of evolution, the wonders of nature, our more primal
selves! To our success!' The bewildered guests pause before drinking the toast. 'Now, let
me prepare for the surprise of the evening!' Dr. Zilch says as he suddenly exits the
ballroom.\n";
451         break;

```



```

452     case 2:
453         retString = "First mate Pole bursts into the ballroom. 'Captain Swell! The
lifeboats! Someone has cast them all off!' 'What?! Well, for now, I suggest all our
guests return to the lounge and wait, if you please.' And so ...\n";
454         break;
455     case 32:
456         retString = "A bright light shines over the water; a spotlight from a coast
guard cruiser. 'Attention! Be prepared to be boarded! This is the coast guard and we're
here to help! Anyone holding weapons will be considered hostile! Stay calm and cooperate
with us!' The cruiser comes alongside and holds long enough for a small team of armed men
to board the ship.\n";
457         break;
458     case 33:
459         retString = "A small uniformed coast guard team boards the ship ... And you
hope that there are enough of them to handle any remaining zombies. Because if you missed
some, and this infection spreads further, the whole world could be at risk. Fingers
crossed ...\n";
460         break;
461     default:
462         // no story
463         break;
464     }
465     retString = InsertNewlines(retString, 79);
466     return retString;
467 }
468
469 string LocationFormat( int locIndex ) {
470     string retString = "";
471     switch (locIndex) {
472     case 0:
473         retString = "\n[BRIDGE]";
474         break;
475     case 1:
476         retString = "\n[FORE DECK]";
477         break;
478     case 2:
479         retString = "\n[AFT DECK]";
480         break;
481     case 3:
482         retString = "\n[BALLROOM]";
483         break;
484     case 4:
485         retString = "\n[LOUNGE]";
486         break;
487     case 5:
488         retString = "\n[KITCHEN]";
489         break;
490     case 6:
491         retString = "\n[STORE ROOM]";
492         break;
493     case 7:
494         retString = "\n[ENGINE ROOM]";
495         break;
496     }
497     return retString;
498 }
499
500 string LocationDescriptionFormat( int locIndex ) {
501     string retString = "";
502     switch (locIndex) {
503     case 0:
504         retString = "\n[BRIDGE]";
505         retString += "\nAt the highest point on the ship, the bridge is the command
center with all\nship controls.";
506         retString += "\nThere is a radio here, with emergency instructions for
contacting the coast\nguard.";

```

```

507         retString += "\nThere is a first aid kit here, with simple instructions to
heal a variety of\nwounds.";
508         retString += "\nFrom here, there is an exit to the Fore deck, the Aft deck
and the Lounge.";
509         break;
510     case 1:
511         retString = "\n[FORE DECK]";
512         retString += "\nThis is a large open deck in the front of the ship and
above the waves\nbelow.";
513         retString += "\nFrom here, there is an exit to the Bridge, the Ballroom and
the Store room.";
514         break;
515     case 2:
516         retString = "\n[AFT DECK]";
517         retString += "\nThis is an open air deck at the back of the ship with a
view of the moon\nshining on the ocean.";
518         retString += "\nFrom here, there is an exit to the Bridge, the Lounge and
the Engine room.";
519         break;
520     case 3:
521         retString = "\n[BALLROOM]";
522         retString += "\nThis is an elaborate dancing and dining ballroom the serves
as the primary\ngathering place on the ship.";
523         retString += "\nFrom here, there is an exit to the Fore deck, the Lounge
and the Kitchen.";
524         break;
525     case 4:
526         retString = "\n[LOUNGE]";
527         retString += "\nThis small dimly lit lounge is fit for mixing and mingling
with other guests\non board.";
528         retString += "\nFrom here, there is an exit to the Bridge, the Aft deck and
the Ballroom.";
529         break;
530     case 5:
531         retString = "\n[KITCHEN]";
532         retString += "\nThis is a large galley kitchen capable of serving a few
dozen people at a time.";
533         retString += "\nThere is a first aid kit here, with simple instructions to
heal a variety of\nwounds.";
534         retString += "\nFrom here, there is an exit to the Ballroom, the Store room
and the Engine\nroom.";
535         break;
536     case 6:
537         retString = "\n[STORE ROOM]";
538         retString += "\nThis is a utility storage room used for supplies and
maintenance.";
539         retString += "\nFrom here, there is an exit to the Kitchen, the Engine room
and the Fore deck.";
540         break;
541     case 7:
542         retString = "\n[ENGINE ROOM]";
543         retString += "\nThis is the main engine room of the ship with a large
diesel engine\ndominating the room.";
544         retString += "\nFrom here, there is an exit to the Kitchen, the Store room
and the Aft deck.";
545         break;
546     }
547     return retString;
548 }
549
550 string ItemName( int itemIndex ) {
551     string retString = "";
552     switch (itemIndex) {
553     case 0:
554         retString = "a flare gun";
555         break;

```

```

556     case 1:
557         retString = "a fire extinguisher";
558         break;
559     case 2:
560         retString = "an alcohol bottle";
561         break;
562     case 3:
563         retString = "a diving knife";
564         break;
565     case 4:
566         retString = "a spear gun";
567         break;
568     case 5:
569         retString = "a wrench";
570         break;
571     case 6:
572         retString = "a cleaver";
573         break;
574     case 7:
575         retString = "a fuel can";
576         break;
577     }
578     return retString;
579 }
580
581 string ItemFormat( int itemIndex ) {
582     string retString = "There is ";
583     retString += ItemName( itemIndex );
584     retString += " here.";
585     return retString;
586 }
587
588 string ItemDescriptionFormat( int itemIndex ) {
589     string retString = "";
590     switch (itemIndex) {
591     case 0:
592         retString = "This hand-held flare gun is used to signal other ships with a
magnesium flare.";
593         break;
594     case 1:
595         retString = "This full-sized metal fire extinguisher is used to put out
medium and small fires.";
596         break;
597     case 2:
598         retString = "This is a full bottle of premium spirit alcohol.";
599         break;
600     case 3:
601         retString = "This is a large hand-held blade used during diving for utility
and safety.";
602         break;
603     case 4:
604         retString = "This is a rifle-sized compressed air spear gun used in
emergencies while diving.";
605         break;
606     case 5:
607         retString = "This is a very large monkey wrench used during maintenance of
heavy machinery.";
608         break;
609     case 6:
610         retString = "This large meat cleaver is very sharp and used to de-bone and
section large cuts of meat.";
611         break;
612     case 7:
613         retString = "This two gallon fuel can is used to hold diesel fuel in
reserve.";
614         break;

```

```

615     }
616     retString = InsertNewlines(retString, 79);
617     return retString;
618 }
619
620 string CharacterDescriptionFormat( int charIndex ) {
621     string retString = "";
622     switch (charIndex) {
623     case 0:
624         retString = "Captain Swell is a heavy set man with a thick salt and pepper
beard. He wears the uniform of a ship's captain.";
625         break;
626     case 1:
627         retString = "First mate Pole is a tall thin man with particularly good
posture. He wears the uniform of a crewman.";
628         break;
629     case 2:
630         retString = "Chef Rotisserie is a stout barrel-chested man with a long
mustache and dark hair arranged in a comb-over.";
631         break;
632     case 3:
633         retString = "Phil is the ship's bartender. He has tied his hair back in a
short ponytail, and wears a red vest.";
634         break;
635     case 4:
636         retString = "Mr. Rich is an elderly gentleman with short white hair, and
wearing a fine blue suit with a red tie.";
637         break;
638     case 5:
639         retString = "Mrs. Rich is a sophisticated woman with a long yellow evening
gown and a red flower in her hair.";
640         break;
641     case 6:
642         retString = "Prof. Smart is a tall man in wire-rimmed glasses, wearing a
simple brown wool suit.";
643         break;
644     case 7:
645         retString = "Ms. Sass is a short thin young woman with long black hair and
glasses, wearing a dark purple dress.";
646         break;
647     }
648     retString = InsertNewlines(retString, 79);
649     return retString;
650 }
651
652 string CharacterName( int charIndex, bool zombie ) {
653     string retString = "";
654     if ( zombie )
655         retString = "Zombie ";
656     switch (charIndex) {
657     case 0:
658         retString += "Captain Swell";
659         break;
660     case 1:
661         retString += "First Mate Pole";
662         break;
663     case 2:
664         retString += "Chef Rotisserie";
665         break;
666     case 3:
667         retString += "Phil";
668         break;
669     case 4:
670         retString += "Mr. Rich";
671         break;
672     case 5:

```

```

673         retString += "Mrs. Rich";
674         break;
675     case 6:
676         retString += "Prof. Smart";
677         break;
678     case 7:
679         retString += "Ms. Sass";
680     }
681     return retString;
682 }
683
684 string CharacterDialogResponse( int charIndex ) {
685     string retString = "";
686     int r = (rand() % 2);
687     switch (charIndex) {
688     case 0:
689         if ( r == 0 )
690             retString = "Ahoy there";
691         else
692             retString = "If you need anything, just ask";
693         break;
694     case 1:
695         if ( r == 0 )
696             retString = "Here to help";
697         else
698             retString = "At your service";
699         break;
700     case 2:
701         if ( r == 0 )
702             retString = "Stay out of my kitchen";
703         else
704             retString = "Bon appetit";
705         break;
706     case 3:
707         if ( r == 0 )
708             retString = "It's always happy hour";
709         else
710             retString = "I'm here to listen";
711         break;
712     case 4:
713         if ( r == 0 )
714             retString = "Bully!";
715         else
716             retString = "I thought this was a private cruise";
717         break;
718     case 5:
719         if ( r == 0 )
720             retString = "Well, I never!";
721         else
722             retString = "The service here is awful";
723         break;
724     case 6:
725         if ( r == 0 )
726             retString = "This is fascinating";
727         else
728             retString = "I have a theory";
729         break;
730     case 7:
731         if ( r == 0 )
732             retString = "Seriously?";
733         else
734             retString = "Worst cruise ever";
735         break;
736     }
737     return retString;
738 }

```

```

739
740 string CharacterDialogQuestion( int charIndex ) {
741     string retString = "";
742     switch (charIndex) {
743     case 0:
744         retString = "Is everyone all right?";
745         break;
746     case 1:
747         retString = "Can I get anyone anything?";
748         break;
749     case 2:
750         retString = "Is anyone hungry?";
751         break;
752     case 3:
753         retString = "Another round?";
754         break;
755     case 4:
756         retString = "Do you know how much money I have?";
757         break;
758     case 5:
759         retString = "Can you help me?";
760         break;
761     case 6:
762         retString = "Don't you see what this means?";
763         break;
764     case 7:
765         retString = "Could this be any more tragic?";
766         break;
767     }
768     return retString;
769 }
770
771 string CharacterDialogAnswer( int charIndex ) {
772     string retString = "";
773     int r = (rand() % 2);
774     switch (charIndex) {
775     case 0:
776         if ( r == 0 )
777             retString = "Certainly";
778         else
779             retString = "I'm afraid not";
780         break;
781     case 1:
782         if ( r == 0 )
783             retString = "Yes";
784         else
785             retString = "Not really";
786         break;
787     case 2:
788         if ( r == 0 )
789             retString = "Oui oui";
790         else
791             retString = "No";
792         break;
793     case 3:
794         if ( r == 0 )
795             retString = "Yeah";
796         else
797             retString = "Nope";
798         break;
799     case 4:
800         if ( r == 0 )
801             retString = "Of course";
802         else
803             retString = "Not at all";
804         break;

```

```

805     case 5:
806         if ( r == 0 )
807             retString = "Yes indeed";
808         else
809             retString = "Certainly not";
810         break;
811     case 6:
812         if ( r == 0 )
813             retString = "I think so";
814         else
815             retString = "It's unknowable";
816         break;
817     case 7:
818         if ( r == 0 )
819             retString = "Whatever";
820         else
821             retString = "Like I care";
822         break;
823 }
824 return retString;
825 }
826
827 string CharacterDialogExclamation( int charIndex ) {
828     string retString = "";
829     switch (charIndex) {
830     case 0:
831         retString = "Oh!";
832         break;
833     case 1:
834         retString = "Ah!";
835         break;
836     case 2:
837         retString = "Sacre bleu!";
838         break;
839     case 3:
840         retString = "Hey!";
841         break;
842     case 4:
843         retString = "I say!";
844         break;
845     case 5:
846         retString = "My stars!";
847         break;
848     case 6:
849         retString = "Stay back!";
850         break;
851     case 7:
852         retString = "Eww!";
853         break;
854     }
855     return retString;
856 }
857
858 string CharacterDialogResting( int charIndex ) {
859     string retString = "";
860     switch (charIndex) {
861     case 0:
862         retString = "I need to rest";
863         break;
864     case 1:
865         retString = "Let me sit down";
866         break;
867     case 2:
868         retString = "Excuse moi";
869         break;
870     case 3:

```

```

871         retString = "I'm going down";
872         break;
873     case 4:
874         retString = "I'm all right";
875         break;
876     case 5:
877         retString = "I feel faint";
878         break;
879     case 6:
880         retString = "I need a minute";
881         break;
882     case 7:
883         retString = "Just leave me alone";
884         break;
885     }
886     return retString;
887 }
888
889 string CharacterDialogZombie( int charIndex ) {
890     string retString = "";
891     switch (charIndex) {
892     case 0:
893         retString = "Ahhuugh thuuugh";
894         break;
895     case 1:
896         retString = "Huungh tuugh huuughp";
897         break;
898     case 2:
899         retString = "Buughn appuughtuuught";
900         break;
901     case 3:
902         retString = "Uuhts uuhlwuhs huuuhpugh huugh";
903         break;
904     case 4:
905         retString = "Buuulluughe";
906         break;
907     case 5:
908         retString = "Wuuulh uuh nuuvvuugh";
909         break;
910     case 6:
911         retString = "Thuughs ughs fuuusuunughtugh";
912         break;
913     case 7:
914         retString = "Surunghsluughee";
915         break;
916     }
917     return retString;
918 }
919
920 string ZCGame::InventoryFormat() {
921     string retString = "\n You have ";
922     int itemCount = 0;
923     for ( int i=0; i<12; i++ ) {
924         if ( GetBit(items[i].itemBits, (1<<1)) ) {
925             itemCount++;
926             retString += "\n";
927             switch ( GetNum(items[i].itemBits,5,3) ) {
928             case 0:
929                 retString += " a flare gun";
930                 break;
931             case 1:
932                 retString += " a fire extinguisher";
933                 break;
934             case 2:
935                 retString += " an alcohol bottle";
936                 break;

```



```

937         case 3:
938             retString += "  a diving knife";
939             break;
940         case 4:
941             retString += "  a spear gun";
942             break;
943         case 5:
944             retString += "  a wrench";
945             break;
946         case 6:
947             retString += "  a cleaver";
948             break;
949         case 7:
950             retString += "  a fuel can";
951             break;
952     }
953 }
954 }
955 if ( itemCount == 0 ) {
956     retString += "nothing of use.";
957 }
958 // equipped items
959 int equipItemA = GetNum( player.pBits, 7, 4 );
960 int equipItemB = GetNum( player.pBits, 3, 4 );
961 if ( equipItemA < 12 ) {
962     retString += "\n You hold " + ItemName( GetNum( items[equipItemA].itemBits,
5, 3 ) ) + " in your right hand";
963 }
964 if ( equipItemB < 12 ) {
965     retString += "\n You hold " + ItemName( GetNum( items[equipItemB].itemBits,
5, 3 ) ) + " in your left hand";
966 }
967 return retString;
968 }
969
970 bool ZCGame::TakeItem( int itemIndex ) {
971     bool retBool = false;
972     int inventoryCount = GetNum( player.pBits, 0, 3 );
973     // if item is not taken and item location is player location, take
974     if ( !GetBit( items[itemIndex].itemBits, (1<<1) ) && ( GetNum(items[itemIndex].
itemBits, 2, 3) == GetNum(player.pBits, 13, 3) ) ) {
975         if ( GetBit( items[itemIndex].itemBits, (1<<0) ) ) {
976             // used items cannot be taken
977         }
978         else if ( inventoryCount < 7 ) {
979             // taken if ( inventory count < 7 )
980             items[itemIndex].itemBits = SetBit( items[itemIndex].itemBits, true, (1
<<1) );
981             retBool = true;
982             // increment inventory count if taken
983             inventoryCount++;
984             player.pBits = SetNum( player.pBits, inventoryCount, 0, 3 );
985         }
986         else {
987             cout << "\nYou hold too much, so " << ItemName( GetNum(items[itemIndex
].itemBits, 5, 3) ) << " is not taken.";
988         }
989     }
990     return retBool;
991 }
992
993 bool ZCGame::DropItem( int itemIndex ) {
994     bool retBool = false;
995     int itemType = GetNum( items[itemIndex].itemBits, 5, 3 );
996     if ( GetBit( items[itemIndex].itemBits, (1<<1) ) ) {
997         // if equipped, un-equip (set to value 15)

```

```

998         if ( itemIndex == GetNum( player.pBits, 7, 4 ) ) {
999             player.pBits = SetNum( player.pBits, 15, 7, 4 );
1000             cout << "\nNow " << ItemName( itemType ) << " is no longer equipped in
your right hand.";
1001         }
1002         else if ( itemIndex == GetNum( player.pBits, 3, 4 ) ) {
1003             player.pBits = SetNum( player.pBits, 15, 3, 4 );
1004             cout << "\nNow " << ItemName( itemType ) << " is no longer equipped in
your left hand.";
1005         }
1006         // drop here
1007         items[itemIndex].itemBits = SetBit( items[itemIndex].itemBits, false, (1<<1
) );
1008         items[itemIndex].itemBits = SetNum( items[itemIndex].itemBits, GetNum(
player.pBits, 13, 3 ), 2, 3 );
1009         retBool = true;
1010         // decrement inventory count
1011         int inventoryCount = GetNum( player.pBits, 0, 3 );
1012         inventoryCount--;
1013         player.pBits = SetNum( player.pBits, inventoryCount, 0, 3 );
1014     }
1015     return retBool;
1016 }
1017
1018 void ZCGame::EquipItem( int itemType ) {
1019     int itemIndexA = GetNum( player.pBits, 7, 4 );
1020     int itemIndexB = GetNum( player.pBits, 3, 4 );
1021     // check for items matching type in inventory
1022     int itemIndex = FindItemInInventory( itemType );
1023     if ( itemIndex > -1 ) {
1024         if ( GetBit( items[itemIndex].itemBits, (1<<1) ) && itemIndexA != itemIndex
&& itemIndexB != itemIndex ) {
1025             // items held and not already equipped
1026             if ( itemIndexA == 15 ) {
1027                 itemIndexA = itemIndex;
1028                 items[itemIndex].itemBits = SetBit( items[itemIndex].itemBits, true
, (1<<1) );
1029                 player.pBits = SetNum( player.pBits, itemIndex, 7, 4 );
1030                 cout << "\nNow " << ItemName( GetNum( items[itemIndexA].itemBits, 5,
3 ) ) << " is equipped in your right hand.";
1031             }
1032             else if ( itemIndexB == 15 ) {
1033                 itemIndexB = itemIndex;
1034                 items[itemIndex].itemBits = SetBit( items[itemIndex].itemBits, true
, (1<<1) );
1035                 player.pBits = SetNum( player.pBits, itemIndex, 3, 4 );
1036                 cout << "\nNow " << ItemName( GetNum( items[itemIndexB].itemBits, 5,
3 ) ) << " is equipped in your left hand.";
1037             }
1038             else {
1039                 // un-equip left hand item
1040                 items[itemIndexB].itemBits = SetBit( items[itemIndexB].itemBits,
false, (1<<1) );
1041                 cout << "\nYou keep " << ItemName( GetNum( items[itemIndexB].
itemBits, 5, 3 ) ) << ", but ...";
1042                 // switch right hand item to left
1043                 itemIndexB = itemIndexA;
1044                 player.pBits = SetNum( player.pBits, itemIndexB, 3, 4 );
1045                 // equip new item right hand
1046                 itemIndexA = itemIndex;
1047                 player.pBits = SetNum( player.pBits, itemIndexA, 7, 4 );
1048                 cout << "\nIn your right hand, you now hold " << ItemName( GetNum(
items[itemIndexA].itemBits, 5, 3 ) ) << " ...";
1049                 cout << "\nand " << ItemName( GetNum( items[itemIndexB].itemBits, 5,
3 ) ) << " is equipped in your left hand.";
1050             }

```

```

1051     }
1052     else if ( itemIndexA == itemIndex ) {
1053         cout << "\nYou already have that item equipped in your right hand.";
1054     }
1055     else if ( itemIndexB == itemIndex ) {
1056         cout << "\nYou already have that item equipped in your left hand.";
1057     }
1058 }
1059 else {
1060     cout << "\nYou do not have that item.";
1061 }
1062 }
1063
1064 void ZCGame::EquipAny() {
1065     int itemIndexA = GetNum( player.pBits, 7, 4 );
1066     int itemIndexB = GetNum( player.pBits, 3, 4 );
1067     for ( int i=0; i<12; i++ ) {
1068         if ( GetBit( items[i].itemBits, (1<<1) ) && itemIndexA != i && itemIndexB
1069 != i ) {
1070             // items held and not already equipped
1071             if ( itemIndexA == 15 ) {
1072                 itemIndexA = i;
1073                 items[i].itemBits = SetBit( items[i].itemBits, true, (1<<1) );
1074                 player.pBits = SetNum( player.pBits, i, 7, 4 );
1075                 cout << "\nNow " << ItemName( GetNum( items[itemIndexA].itemBits, 5,
1076 3 )) << " is equipped in your right hand.";
1077             }
1078             else if ( itemIndexB == 15 ) {
1079                 itemIndexB = i;
1080                 items[i].itemBits = SetBit( items[i].itemBits, true, (1<<1) );
1081                 player.pBits = SetNum( player.pBits, i, 3, 4 );
1082                 cout << "\nNow " << ItemName( GetNum( items[itemIndexB].itemBits, 5,
1083 3 )) << " is equipped in your left hand.";
1084             }
1085         }
1086     }
1087 }
1088
1089 bool ZCGame::UseItem( int itemIndex ) {
1090     bool retBool = false;
1091     // if one-use item, use and return true
1092     int itemType = GetNum( items[itemIndex].itemBits, 5, 3 );
1093     if ( itemType == 0 || itemType == 1 || itemType == 2 || itemType == 4 ||
1094 itemType == 7 ) {
1095         if ( itemType != 1 ) {
1096             // all except fire extinguisher can only be used once
1097             items[itemIndex].itemBits = SetBit( items[itemIndex].itemBits, true, (1
1098 <<0) );
1099         }
1100         // if alcohol bottle or fuel can, make current location flammable
1101         int currRoom = GetNum( player.pBits, 13, 3 ); // current room is where
1102 player is
1103         items[itemIndex].itemBits = SetNum( items[itemIndex].itemBits, currRoom, 2,
1104 3 ); // set item location here
1105         if ( itemType == 2 || itemType == 7 ) {
1106             if ( GetNum( locations[currRoom].locBits, 4, 2 ) == 2 ) {
1107                 // on fire area resets fire timer
1108                 cout << "\n ... and fuel is added to the fire.";
1109                 locations[ currRoom ].locBits = SetNum( locations[ currRoom ].
1110 locBits, 1, 0, 4 ); // fire timer reset
1111             }
1112             else {
1113                 cout << "\n ... flammable liquid spills everywhere.";
1114                 locations[ currRoom ].locBits = SetNum( locations[ currRoom ].
1115 locBits, 1, 4, 2 ); // room flammable
1116             }
1117         }
1118     }
1119 }

```

```

1108     }
1109     else if ( itemType == 1 ) {
1110         // fire extinguisher puts out fire
1111         if ( GetNum( locations[currRoom].locBits, 4, 2 ) == 2 ) {
1112             // on fire area is extinguished
1113             cout << "\nWith some work, the fire is extinguished.";
1114             locations[ currRoom ].locBits = SetNum( locations[ currRoom ].
locBits, 3, 4, 2 ); // room burnt
1115         }
1116         else {
1117             cout << "\nWhite fire retardant coats the area, and slowly
disappears.";
1118         }
1119     }
1120     else if ( itemType == 0 ) {
1121         // flare gun ignites flammable areas
1122         if ( GetNum( locations[currRoom].locBits, 4, 2 ) == 1 ) {
1123             cout << "\n ... and the area bursts into flames.";
1124             locations[ currRoom ].locBits = SetNum( locations[ currRoom ].
locBits, 2, 4, 2 ); // room flammable
1125             locations[ currRoom ].locBits = SetNum( locations[ currRoom ].
locBits, 1, 0, 4 ); // fire timer reset
1126         }
1127     }
1128     retBool = true;
1129 }
1130 return retBool;
1131 }
1132
1133 void ZCGame::InfectCharacter( int charIndex ) {
1134     characters[charIndex].charBits = SetBit( characters[charIndex].charBits, true,
(1<<1) );
1135     // force immobile
1136     characters[charIndex].charBits = SetBit( characters[charIndex].charBits, true,
(1<<4) );
1137 }
1138
1139 string PlayerPrompt() {
1140     cout << "[Your Turn] > ";
1141     string playerMove;
1142     getline(cin, playerMove );
1143     return playerMove;
1144 }
1145
1146 string* WordBreak( string s ) {
1147     string* retStrings = new string[5];
1148     int wordCount = 0;
1149     string workS = s;
1150     string currWord = "";
1151     int wordLen = 0;
1152     while ( wordLen != workS.npos ) {
1153         wordLen = workS.find( " ", 0 );
1154         currWord = workS.substr( 0, wordLen );
1155         workS = workS.substr( (wordLen+1), workS.npos );
1156         retStrings[ wordCount++ ] = currWord;
1157     }
1158     return retStrings;
1159 }
1160
1161 int ZCGame::DisambiguateLocation( string* w ) {
1162     int retInt = -1;
1163     // Locations: [8] Bridge, Fore Deck, Aft Deck, Ballroom, Lounge, Kitchen, Store
Room, Engine Room
1164     if ( w[0] == "Bridge" || w[0] == "bridge" || w[1] == "Bridge" || w[1] ==
"bridge" ) {
1165         retInt = 0;

```

```

1166     }
1167     else if ( w[0] == "Foredeck" || w[0] == "foredeck" || w[1] == "Foredeck" || w[1]
] == "foredeck" || w[0] == "Fore" || w[0] == "fore" || w[1] == "Fore" || w[1] == "fore" )
{
1168         retInt = 1;
1169     }
1170     else if ( w[0] == "Aftdeck" || w[0] == "aftdeck" || w[1] == "Aftdeck" || w[1]
== "aftdeck" || w[0] == "Aft" || w[0] == "aft" || w[1] == "Aft" || w[1] == "aft" ) {
1171         retInt = 2;
1172     }
1173     else if ( w[0] == "Ballroom" || w[0] == "ballroom" || w[1] == "Ballroom" || w[1]
] == "ballroom" || w[0] == "Ball" || w[0] == "ball" || w[1] == "Ball" || w[1] == "ball" )
{
1174         retInt = 3;
1175     }
1176     else if ( w[0] == "Lounge" || w[0] == "lounge" || w[1] == "Lounge" || w[1] ==
"lounge" ) {
1177         retInt = 4;
1178     }
1179     else if ( w[0] == "Kitchen" || w[0] == "kitchen" || w[1] == "Kitchen" || w[1]
== "kitchen" ) {
1180         retInt = 5;
1181     }
1182     else if ( w[0] == "Storeroom" || w[0] == "storeroom" || w[1] == "Storeroom" ||
w[1] == "storeroom" || w[0] == "Store" || w[0] == "store" || w[1] == "Store" || w[1] ==
"store" ) {
1183         retInt = 6;
1184     }
1185     else if ( w[0] == "Engineroom" || w[0] == "engineroom" || w[1] == "Engineroom"
|| w[1] == "engineroom" || w[0] == "Engine" || w[0] == "engine" || w[1] == "Engine" || w[
1] == "engine" ) {
1186         retInt = 7;
1187     }
1188     return retInt;
1189 }
1190
1191 int ZCGame::DisambiguateItem( string* w ) {
1192     int retInt = -1;
1193     // Items: [8] Flare Gun, Fire Extinguisher, Alcohol Bottle, Diving Knife, Spear
Gun, Wrench, Cleaver, Fuel Can
1194     if ( w[0] == "Flare" || w[0] == "flare" || w[1] == "Flare" || w[1] == "flare"
|| w[0] == "Flaregun" || w[0] == "flaregun" || w[1] == "Flaregun" || w[1] == "flaregun" )
{
1195         retInt = 0;
1196     }
1197     else if ( w[0] == "Fire" || w[0] == "fire" || w[1] == "Fire" || w[1] == "fire"
|| w[0] == "Extinguisher" || w[0] == "extinguisher" || w[1] == "Extinguisher" || w[1] ==
"extinguisher" ) {
1198         retInt = 1;
1199     }
1200     else if ( w[0] == "Alcohol" || w[0] == "alcohol" || w[1] == "Alcohol" || w[1]
== "alcohol" || w[0] == "Bottle" || w[0] == "bottle" || w[1] == "Bottle" || w[1] ==
"bottle" ) {
1201         retInt = 2;
1202     }
1203     else if ( w[0] == "Diving" || w[0] == "diving" || w[1] == "Diving" || w[1] ==
"diving" || w[0] == "Knife" || w[0] == "knife" || w[1] == "Knife" || w[1] == "knife" ) {
1204         retInt = 3;
1205     }
1206     else if ( w[0] == "Spear" || w[0] == "spear" || w[1] == "Spear" || w[1] ==
"spear" || w[0] == "Speargun" || w[0] == "speargun" || w[1] == "Speargun" || w[1] ==
"speargun" ) {
1207         retInt = 4;
1208     }
1209     else if ( w[0] == "Wrench" || w[0] == "wrench" || w[1] == "Wrench" || w[1] ==
"wrench" ) {

```

```

1210         retInt = 5;
1211     }
1212     else if ( w[0] == "Cleaver" || w[0] == "cleaver" || w[1] == "Cleaver" || w[1]
== "cleaver" ) {
1213         retInt = 6;
1214     }
1215     else if ( w[0] == "Fuel" || w[0] == "fuel" || w[1] == "Fuel" || w[1] == "fuel"
|| w[0] == "Can" || w[0] == "can" || w[1] == "Fuelcan" || w[1] == "fuelcan" ) {
1216         retInt = 7;
1217     }
1218     return retInt;
1219 }
1220
1221 int ZCGame::DisambiguateCharacter( string* w ) {
1222     int retInt = -1;
1223     // People: [8] Captain, 1st Mate, Chef, Bartender, Mr. Rich, Mrs. Rich, Prof.
Smart, Ms. Sass
1224     if ( w[0] == "Swell" || w[0] == "swell" || w[1] == "Swell" || w[1] == "swell" )
{
1225         retInt = 0;
1226     }
1227     else if ( w[0] == "Pole" || w[1] == "Pole" || w[2] == "Pole" || w[0] == "pole"
|| w[1] == "pole" || w[2] == "pole" ) {
1228         retInt = 1;
1229     }
1230     else if ( w[0] == "Chef" || w[1] == "chef" || w[1] == "Rotisserie" || w[0] ==
"chef" || w[1] == "chef" || w[1] == "rotisserie" || w[2] == "rotisserie" ) {
1231         retInt = 2;
1232     }
1233     else if ( w[0] == "Phil" || w[0] == "phil" || w[1] == "Phil" || w[1] == "phil"
) {
1234         retInt = 3;
1235     }
1236     else if ( ( w[0] == "Mr." || w[0] == "mr." || w[0] == "mr" ) && ( w[1] ==
"Rich" || w[1] == "rich" ) ) {
1237         retInt = 4;
1238     }
1239     else if ( ( w[0] == "Mrs." || w[0] == "mrs." || w[0] == "mrs" ) && ( w[1] ==
"Rich" || w[1] == "rich" ) ) {
1240         retInt = 5;
1241     }
1242     else if ( w[0] == "Smart" || w[0] == "smart" || w[1] == "Smart" || w[1] ==
"smart" ) {
1243         retInt = 6;
1244     }
1245     else if ( w[0] == "Sass" || w[0] == "sass" || w[1] == "Sass" || w[1] == "sass"
) {
1246         retInt = 7;
1247     }
1248     return retInt;
1249 }
1250
1251 int ZCGame::FindItemInLocation( int itemType ) {
1252     // return index of item among all including additional random items (8-11)
1253     int retInt = -1; // -1 = not found
1254     for (int i=0; i<12; i++) {
1255         if ( GetNum( items[i].itemBits, 5, 3 ) == itemType ) {
1256             if ( GetNum( player.pBits, 13, 3 ) == GetNum( items[i].itemBits, 2, 3 )
) {
1257                 // exclude used items
1258                 if ( !GetBit( items[i].itemBits, (1<<0) ) ) {
1259                     retInt = i;
1260                     break;
1261                 }
1262             }
1263             else {
1264                 retInt = -2; // -2 = found in location but used (feedback to

```

```

player)
1264         }
1265     }
1266 }
1267 }
1268 return retInt;
1269 }
1270
1271 int ZCGame::FindItemInInventory( int itemType ) {
1272     // return index of item among all including additional random items (8-11)
1273     int retInt = -1;
1274     for (int i=0; i<12; i++) {
1275         if ( GetNum( items[i].itemBits, 5, 3 ) == itemType ) {
1276             if ( GetBit( items[i].itemBits, (1<<1) ) ) {
1277                 retInt = i;
1278                 break;
1279             }
1280         }
1281     }
1282     return retInt;
1283 }
1284
1285 int FindRoomFromExit( int currentRoom, int exitIndex ) {
1286     // return room index from current room and exit index
1287     int retInt = currentRoom;
1288     // Exits: 0=1-2-4, 1=0-3-6, 2=0-4-7, 3=1-4-5, 4=0-2-3, 5=3-6-7, 6=1-5-7,
1289     // 7=2-5-6
1290     switch (currentRoom) {
1291     case 0:
1292         if ( exitIndex == 0 )
1293             retInt = 1;
1294         else if ( exitIndex == 1 )
1295             retInt = 2;
1296         else
1297             retInt = 4;
1298         break;
1299     case 1:
1300         if ( exitIndex == 0 )
1301             retInt = 0;
1302         else if ( exitIndex == 1 )
1303             retInt = 3;
1304         else
1305             retInt = 6;
1306         break;
1307     case 2:
1308         if ( exitIndex == 0 )
1309             retInt = 0;
1310         else if ( exitIndex == 1 )
1311             retInt = 4;
1312         else
1313             retInt = 7;
1314         break;
1315     case 3:
1316         if ( exitIndex == 0 )
1317             retInt = 1;
1318         else if ( exitIndex == 1 )
1319             retInt = 4;
1320         else
1321             retInt = 5;
1322         break;
1323     case 4:
1324         if ( exitIndex == 0 )
1325             retInt = 0;
1326         else if ( exitIndex == 1 )
1327             retInt = 2;
1328         else

```

```

1328         retInt = 3;
1329     break;
1330 case 5:
1331     if ( exitIndex == 0 )
1332         retInt = 3;
1333     else if ( exitIndex == 1 )
1334         retInt = 6;
1335     else
1336         retInt = 7;
1337     break;
1338 case 6:
1339     if ( exitIndex == 0 )
1340         retInt = 1;
1341     else if ( exitIndex == 1 )
1342         retInt = 5;
1343     else
1344         retInt = 7;
1345     break;
1346 case 7:
1347     if ( exitIndex == 0 )
1348         retInt = 2;
1349     else if ( exitIndex == 1 )
1350         retInt = 5;
1351     else
1352         retInt = 6;
1353     break;
1354 }
1355 return retInt;
1356 }
1357
1358 void ZCGame::ParseMove( string pMove ) {
1359     // break move into words (5 words max)
1360     string* word = new string[5];
1361     word = WordBreak( pMove );
1362     // match to commands
1363     // identify command targets
1364     if ( word[0] == "debug" ) {
1365         if ( word[1] == "game" ) {
1366             cout << DebugBits( "gameBits", gameBits );
1367         }
1368         else if ( word[1] == "score" ) {
1369             cout << DebugBits( "scoreBits", scoreBits );
1370         }
1371         else if ( word[1] == "player" ) {
1372             cout << DebugBits( "playerBits", player.pBits );
1373         }
1374         else if ( word[1] == "locations" || word[1] == "loc" || word[1] == "locs" )
1375         {
1376             for (int i=0;i<8;i++) {
1377                 cout << DebugBits( "locBits", locations[i].locBits );
1378             }
1379         }
1380         else if ( word[1] == "characters" || word[1] == "char" || word[1] ==
"chars" ) {
1381             for (int i=0;i<8;i++) {
1382                 cout << DebugBits( "charBits", characters[i].charBits );
1383             }
1384         }
1385         else if ( word[1] == "items" || word[1] == "item" ) {
1386             for (int i=0;i<8;i++) {
1387                 cout << DebugBits( "itemBits", items[i].itemBits );
1388             }
1389         }
1390         else {
1391             // debug all
1392             cout << DebugBits( "playerBits", player.pBits );

```



```

1392     cout << DebugBits( "gameBits", gameBits );
1393     cout << DebugBits( "scoreBits", scoreBits );
1394     for (int i=0;i<8;i++) {
1395         cout << DebugBits( "locBits", locations[i].locBits );
1396     }
1397     for (int i=0;i<8;i++) {
1398         cout << DebugBits( "charBits", characters[i].charBits );
1399     }
1400     for (int i=0;i<8;i++) {
1401         cout << DebugBits( "itemBits", items[i].itemBits );
1402     }
1403 }
1404     cout << "\n";
1405 }
1406 else if ( word[0] == "quit" ) {
1407     gameBits = SetBit( gameBits, true, (1<<0) );
1408 }
1409 else if ( word[0] == "cheat" ) {
1410     if ( word[1] == "health" || word[1] == "heal" ) {
1411         if ( GetNum( player.pBits, 11, 2 ) < 3 ) {
1412             player.pBits = SetNum( player.pBits, 3, 11, 2 );
1413             cout << "\n-cheat- You feel much better.";
1414         }
1415     }
1416     else if ( word[1] == "teleport" || word[1] == "tport" ) {
1417         string* checkWords = new string[3];
1418         for (int i=0; i<3; i++) {
1419             checkWords[i] = word[(i+2)];
1420         }
1421         int tLoc = DisambiguateLocation( checkWords );
1422         if ( tLoc > -1 && tLoc != GetNum( player.pBits, 13, 3 ) ) {
1423             player.pBits = SetNum( player.pBits, tLoc, 13, 3 );
1424             cout << "\n-cheat- You have been transported to a new location.";
1425         }
1426     }
1427 }
1428 else if ( word[0] == "help" || word[0] == "h" || word[0] == "?" ) {
1429     cout << HelpFormat();
1430 }
1431 else if ( word[0] == "wait" || word[0] == "..." ) {
1432     cout << "\n...";
1433     IncrementTurn();
1434 }
1435 else if ( word[0] == "look" ) {
1436     if ( word[1] == "at" ) {
1437         if ( word[2] == "me" || word[2] == "myself" ) {
1438             // damage assessment based on player health
1439             switch( GetNum( player.pBits, 11, 2 ) ) {
1440                 case 0:
1441                     cout << "\nYou are dying.";
1442                     break;
1443                 case 1:
1444                     cout << "\nYou are in very rough shape.";
1445                     break;
1446                 case 2:
1447                     cout << "\nYou are hurt, but still okay.";
1448                     break;
1449                 case 3:
1450                     cout << "\nYou are looking good.";
1451                     break;
1452             }
1453             IncrementTurn();
1454         }
1455         else if ( word[2] == "room" || word[2] == "area" ) {
1456             // area look, if lit
1457             if ( !IsDarkArea() ) {

```

```

1458         cout << LocationDescriptionFormat( GetNum( player.pBits, 13, 3
) );
1459     }
1460     else {
1461         cout << "\nIt's too dark here to see.";
1462     }
1463     IncrementTurn();
1464 }
1465 else {
1466     // 'look at' check
1467     int playerLoc = GetNum( player.pBits, 13, 3 );
1468     string* checkWords = new string[3];
1469     for (int i=0; i<3; i++) {
1470         checkWords[i] = word[(i+2)];
1471     }
1472     // char match (must be in area, alive, non-infected and non-zombie)
1473     int charIdx = -1;
1474     charIdx = DisambiguateCharacter( checkWords );
1475     if ( charIdx > -1 ) {
1476         if ( playerLoc == GetNum( characters[charIdx].charBits, 5, 3 )
) {
1477             if ( CharacterAlive(charIdx) && !GetBit( characters[charIdx
].charBits, (1<<1) ) && !GetBit( characters[charIdx].charBits, (1<<0) ) ) {
1478                 cout << "\n" << CharacterDescriptionFormat(charIdx);
1479             }
1480             else {
1481                 cout << "\n" << CharacterName( charIdx, GetBit(
characters[charIdx].charBits, (1<<0) ) ) << " is not looking well.";
1482             }
1483             IncrementTurn();
1484         }
1485         else {
1486             cout << "\nThat person is not here.";
1487         }
1488     }
1489     // check item (must be in inventory or in location)
1490     int itemIdx = -1;
1491     itemIdx = DisambiguateItem( checkWords );
1492     if ( itemIdx > -1 ) {
1493         // check aux items
1494         itemIdx = FindItemInInventory( itemIdx );
1495         if ( itemIdx == -1 ) {
1496             itemIdx = FindItemInLocation( DisambiguateItem( checkWords
) );
1497         }
1498         if ( itemIdx > -1 ) {
1499             cout << "\n" << ItemDescriptionFormat( itemIdx );
1500             IncrementTurn();
1501         }
1502         else {
1503             cout << "\nYou do not see that item.";
1504         }
1505     }
1506     // check location (must be player location -> area look)
1507     int locIdx = -1;
1508     locIdx = DisambiguateLocation( checkWords );
1509     if ( locIdx > -1 ) {
1510         if ( playerLoc == locIdx ) {
1511             cout << LocationDescriptionFormat( GetNum( player.pBits, 13
, 3 ) );
1512             IncrementTurn();
1513         }
1514     }
1515 }
1516 }
1517 else if ( word[1] == "around" || word[1] == "here" ) {

```

```

1518         // area look, if lit
1519         if ( !IsDarkArea() ) {
1520             cout << LocationDescriptionFormat( GetNum( player.pBits, 13, 3 ) );
1521         }
1522         else {
1523             cout << "\nIt's too dark here to see.";
1524         }
1525         IncrementTurn();
1526     }
1527     else {
1528         // area look, if lit
1529         if ( !IsDarkArea() ) {
1530             cout << LocationDescriptionFormat( GetNum( player.pBits, 13, 3 ) );
1531         }
1532         else {
1533             cout << "\nIt's too dark here to see.";
1534         }
1535         IncrementTurn();
1536     }
1537 }
1538 else if ( word[0] == "inventory" || word[0] == "inv" || word[0] == "i" ) {
1539     cout << InventoryFormat();
1540     IncrementTurn();
1541 }
1542 else if ( word[0] == "take" ) {
1543     // 'take' check
1544     int playerLoc = GetNum( player.pBits, 13, 3 );
1545     string* checkWords = new string[4];
1546     for (int i=0; i<4; i++) {
1547         checkWords[i] = word[(i+1)];
1548     }
1549     // item match (must be in area, not used)
1550     int itemIdx = -1;
1551     itemIdx = DisambiguateItem( checkWords );
1552     if ( itemIdx > -1 ) {
1553         // check aux items in location
1554         itemIdx = FindItemInLocation( itemIdx );
1555         if ( itemIdx > -1 ) {
1556             if ( !GetBit( items[itemIdx].itemBits, (1<<1) ) ) {
1557                 if ( TakeItem( itemIdx ) ) {
1558                     cout << "\nYou take " << ItemName( GetNum( items[itemIdx].
itemBits, 5, 3 ) );
1559                     IncrementTurn();
1560                 }
1561             }
1562         }
1563         else if ( itemIdx == -2 ) {
1564             cout << "\nThat item is used and cannot be taken.";
1565         }
1566         else {
1567             cout << "\nThat item is not here.";
1568         }
1569     }
1570     else if ( word[1] == "all" || word[1] == "everything" ) {
1571         for (int i=0; i<12; i++) {
1572             if ( TakeItem( i ) ) {
1573                 cout << "\nYou take " << ItemName( GetNum( items[i].itemBits, 5
, 3 ) );
1574             }
1575         }
1576         IncrementTurn();
1577     }
1578     else {
1579         cout << "\nIt is not clear what item you want to take from here.\n(Name
the item you see here or try 'take all' to take everything)";
1580     }

```

```

1581     }
1582     else if ( word[0] == "drop" ) {
1583         // 'drop' check
1584         string* checkWords = new string[4];
1585         for (int i=0; i<4; i++) {
1586             checkWords[i] = word[(i+1)];
1587         }
1588         // item match (must be in inventory)
1589         int itemIdx = -1;
1590         itemIdx = DisambiguateItem( checkWords );
1591         if ( itemIdx > -1 ) {
1592             // check aux items in inventory
1593             itemIdx = FindItemInInventory( itemIdx );
1594             if ( itemIdx > -1 ) {
1595                 if ( DropItem(itemIdx) ) {
1596                     cout << "\nYou drop " << ItemName( GetNum( items[itemIdx].
itemBits, 5, 3 ) );
1597                     IncrementTurn();
1598                 }
1599             }
1600             else {
1601                 cout << "\nYou do not have that item.";
1602             }
1603         }
1604         else if ( word[1] == "all" || word[1] == "everything" ) {
1605             for (int i=0; i<12; i++) {
1606                 if ( DropItem(i) ) {
1607                     cout << "\nYou drop " << ItemName( GetNum( items[i].itemBits, 5
, 3 ) );
1608                 }
1609             }
1610             IncrementTurn();
1611         }
1612         else {
1613             cout << "\nIt is not clear what item in your inventory you want to
drop.\n(Name the item you have taken or try 'drop all' to drop everything)";
1614         }
1615     }
1616     else if ( word[0] == "equip" ) {
1617         // item match (must be in inventory)
1618         int itemType = -1;
1619         string* checkWords = new string[4];
1620         for (int i=0; i<4; i++) {
1621             checkWords[i] = word[(i+1)];
1622         }
1623         itemType = DisambiguateItem( checkWords );
1624         if ( itemType > -1 ) {
1625             EquipItem( itemType ); // use item type here, EquipItem sorts out if in
inventory
1626             IncrementTurn();
1627         }
1628         else if ( word[1] == "any" || word[1] == "anything" || word[1] ==
"everything" ) {
1629             EquipAny();
1630             IncrementTurn();
1631         }
1632         else {
1633             cout << "\nIt is not clear what item you want to equip in your
hand.\n(Name the item you have taken to equip or try 'equip any' to grab anything)";
1634         }
1635     }
1636     else if ( word[0] == "attack" ) {
1637         DoPlayerAttack();
1638         IncrementTurn();
1639     }
1640     else if ( word[0] == "use" ) {

```

```

1641     int playerLoc = GetNum( player.pBits, 13, 3 );
1642     // item match (must be in inventory, usable type)
1643     string* checkWords = new string[4];
1644     for ( int i=0; i<4; i++ ) {
1645         checkWords[i] = word[(i+1)];
1646     }
1647     int itemType = DisambiguateItem(checkWords);
1648     int itemIdx = FindItemInInventory( itemType );
1649     if ( itemType > -1 && itemIdx > -1 ) {
1650         // use item
1651         if ( !UseItem( itemIdx ) ) {
1652             cout << "\nThere is nothing this item can do.";
1653         }
1654         else if ( itemType != 1 ) {
1655             DropItem( itemIdx ); // single use items are dropped, but not fire
1656             // extinguisher
1657         }
1658         // area match
1659         else if ( word[1] == "radio" ) {
1660             if ( playerLoc == 0 ) {
1661                 if ( GetBit( gameBits, (1<<3) ) ) {
1662                     cout << "\nYou have already called for help and the coast guard
1663 is on the way.";
1664                 }
1665                 else {
1666                     // make sure infection has already been released (infection
1667 timer gets reset as a game balance feature)
1668                     int numDeadChars = 0;
1669                     int numZombiesOnboard = 0;
1670                     int numCharsAlive = 0;
1671                     for ( int i=0; i<8; i++ ) {
1672                         if ( !CharacterAlive(i) ) {
1673                             numDeadChars++;
1674                         }
1675                         else if ( GetBit( characters[i].charBits, (1<<0) ) ) {
1676                             numZombiesOnboard++;
1677                         }
1678                         else {
1679                             numCharsAlive++;
1680                         }
1681                     }
1682                     if ( numDeadChars > 0 || numZombiesOnboard > 0 ) {
1683                         // radio not used, infection released
1684                         gameBits = SetBit( gameBits, true, (1<<3) );
1685                         if ( numCharsAlive > 0 ) {
1686                             cout << "\n YOU [On Radio]: Mayday! Mayday! It's ...
1687 Send armed response! We need help!";
1688                         }
1689                         else {
1690                             cout << "\n YOU [On Radio]: Mayday! Mayday! The ship
1691 ... There's ... Just send help!";
1692                         }
1693                         // skip increment turn (cheat to add a turn before
1694 response)
1695                     }
1696                     else {
1697                         // using radio, but infection not released yet
1698                         cout << "\n Coast Guard [Radio]: This channel is reserved
1699 for emergencies only. Over.";
1700                         IncrementTurn();
1701                     }
1702                 }
1703             }
1704             else {
1705                 cout << "\nThere is no radio here. (Do you think there might be one

```

```

on the bridge?)" ;
1700     }
1701     }
1702     else if ( word[1] == "first" || word[2] == "first" || word[2] == "aid" ||
word[3] == "aid" || word[3] == "kit" || word[4] == "kit" ) {
1703         // first aid kits on the bridge and in the kitchen
1704         if ( playerLoc == 0 || playerLoc == 5 ) {
1705             // first aid kit heals if hurt
1706             if ( GetNum( player.pBits, 11, 2 ) < 3 ) {
1707                 cout << "\nYou're able to heal your wounds somewhat, and you
feel better." ;
1708                 int health = GetNum( player.pBits, 11, 2 );
1709                 player.pBits = SetNum( player.pBits, (health+1), 11, 2 );
1710                 IncrementTurn();
1711             }
1712             else {
1713                 cout << "\nAfter rummaging through the first aid kit, you
decide you're feeling fine." ;
1714                 IncrementTurn();
1715             }
1716         }
1717     }
1718     else {
1719         cout << "\nIt is not clear what you want to use.\n(Name the item you
have taken or the usable feature you see here)" ;
1720     }
1721 }
1722 else if ( word[0] == "talk" ) {
1723     if ( word[1] == "to" ) {
1724         // 'talk to' check
1725         string* checkWords = new string[3];
1726         for (int i=0; i<3; i++) {
1727             checkWords[i] = word[(i+2)];
1728         }
1729         int playerLoc = GetNum( player.pBits, 13, 3 );
1730         int charIdx = -1;
1731         charIdx = DisambiguateCharacter( checkWords );
1732         if ( charIdx > -1 ) {
1733             // char match (must be in area, alive, non-infected and non-zombie)
1734             if ( playerLoc == GetNum( characters[charIdx].charBits, 5, 3 ) &&
CharacterAlive(charIdx) && !GetBit( characters[charIdx].charBits, (1<<1) ) && !GetBit(
characters[charIdx].charBits, (1<<0) ) ) {
1735                 cout << "\n " << CharacterName( charIdx, false ) << ": " <<
CharacterDialogResponse( charIdx );
1736                 IncrementTurn();
1737             }
1738         }
1739         else if ( word[2] == "me" || word[2] == "myself" ) {
1740             cout << "\nNo one notices you are talking to yourself." ;
1741             IncrementTurn();
1742         }
1743         else {
1744             cout << "\nThere is no one here by that name." ;
1745         }
1746     }
1747     else {
1748         cout << "\nNo one notices you are talking to yourself." ;
1749         IncrementTurn();
1750     }
1751 }
1752 else if ( word[0] == "go" || word[0] == "move" || word[0] == "walk" || word[0]
== "run" ) {
1753     if ( word[1] == "to" ) {
1754         string* checkWords = new string[3];
1755         for ( int i=0; i<3; i++ ) {
1756             checkWords[i] = word[(i+2)];

```

```

1757     }
1758     int locIndex = DisambiguateLocation( checkWords );
1759     if ( locIndex > -1 ) {
1760         // location match (must have available exit)
1761         int playerLoc = GetNum( player.pBits, 13, 3 );
1762         int exitA = FindRoomFromExit( playerLoc, 0 );
1763         int exitB = FindRoomFromExit( playerLoc, 1 );
1764         int exitC = FindRoomFromExit( playerLoc, 2 );
1765         if ( exitA == locIndex || exitB == locIndex || exitC == locIndex )
1766         {
1767             player.pBits = SetNum( player.pBits, locIndex, 13, 3 );
1768             IncrementTurn();
1769         }
1770         else {
1771             cout << "\nYou cannot get there from here.\n";
1772             cout << LocationDescriptionFormat( playerLoc );
1773             IncrementTurn();
1774         }
1775     }
1776     else {
1777         cout << "\nThat is not a place you can go to. (Try 'look' to see
1778 the exits from here)";
1779     }
1780     }
1781     else {
1782         cout << "\nTry the command words 'go to' followed by an area, to move
1783 around.\n(or try 'help' for a list of commands)\n";
1784         cout << LocationDescriptionFormat( GetNum( player.pBits, 13, 3 ) );
1785     }
1786     }
1787     else if ( word[0] == "leave" || word[0] == "exit" ) {
1788         cout << "\nTry the command words 'go to' followed by an area, to move
1789 around.\n(or try 'help' for a list of commands)\n";
1790         cout << LocationDescriptionFormat( GetNum( player.pBits, 13, 3 ) );
1791     }
1792     }
1793     else {
1794         // no match
1795         cout << "\nIt's not clear what you mean to do. (Try 'help' for a list of
1796 commands)";
1797         IncrementTurn();
1798     }
1799 }
1800
1801 void ZCGame::IncrementTurn() {
1802     // (skip if game over)
1803     // if not released increment time to release
1804     // check time to release
1805     // if radio used and not S.O.S. call radio respond, flip S.O.S.
1806     // if S.O.S. called, increment turns
1807     // check for rescue time
1808     // if rescue time, call rescue arrived, flip rescue arrived
1809     // check if game over
1810     // Game data: Time to Release (0-7 turns), Is Released, Radio Used, S.O.S.
1811     Called, Rescue Arrived, Game Over
1812     // Time / Scoring data: Time To Rescue (since call for help, 0-31 turns), Score
1813     (0-31)
1814     if ( GetBit( gameBits, (1<<0) ) )
1815         return; // skip if game over
1816     if ( !GetBit( gameBits, (1<<4) ) ) {
1817         int timeToRelease = GetNum( gameBits, 5, 3 );
1818         if ( !GetBit( gameBits, (1<<4) ) ) {
1819             // not yet released
1820             if ( timeToRelease < 7 )
1821                 gameBits = SetNum( gameBits, (timeToRelease+1), 5, 3 );
1822             else {
1823                 // story signal release

```

```

1816         gameBits = SetBit( gameBits, true, (1<<4) );
1817         gameBits = SetNum( gameBits, 2, 5, 3 ); // infection timer reset to
turn #2
1818
1819         // GAME BALANCE NEED (re-start release timer if zombie is killed)
1820         int randChar = ( rand() % 8 );
1821         int safety = 0;
1822         // only alive non-infected and non-zombie character choice
1823         while ( !CharacterAlive(randChar) && safety < 8 && ( GetBit(
characters[randChar].charBits, (1<<1) ) || GetBit( characters[randChar].charBits, (1<<0)
) ) ) {
1824             safety++;
1825             randChar = ( rand() % 8 );
1826         }
1827         if ( safety == 8 ) {
1828             // without valid character selection, reset infection release
timer
1829             gameBits = SetBit( gameBits, false, (1<<4) );
1830         }
1831         else {
1832             // random victim infected
1833             InfectCharacter( randChar );
1834         }
1835     }
1836 }
1837 }
1838 else if ( GetBit( gameBits, (1<<3) ) ) {
1839     if ( !GetBit( gameBits, (1<<2) ) ) {
1840         // story radio respond
1841         gameBits = SetBit( gameBits, true, (1<<2) );
1842         if ( GetNum( player.pBits, 13, 3 ) == 0 ) {
1843             cout << "\n\n Coast Guard [Radio]: We're on our way, just hang
tight. Over.\n";
1844         }
1845     }
1846     else {
1847         // increment time to rescue
1848         int timeToRescue = GetNum( scoreBits, 3, 5 );
1849         if ( timeToRescue < 31 )
1850             scoreBits = SetNum( scoreBits, (timeToRescue+1), 3, 5 );
1851         else {
1852             if ( !GetBit( gameBits, (1<<1) ) ) {
1853                 // story signal rescue arrived
1854                 gameBits = SetBit( gameBits, true, (1<<1) );
1855                 cout << "\n\n" << StoryFormat(32);
1856             }
1857             else {
1858                 // story signal game over
1859                 gameBits = SetBit( gameBits, true, (1<<0) );
1860                 cout << "\n\n" << StoryFormat(33);
1861             }
1862         }
1863     }
1864 }
1865 }
1866
1867 void ZCGame::IncrementScore( int scoreAdd ) {
1868     int oldScore = GetNum( scoreBits, 0, 3 );
1869     scoreBits = SetNum( scoreBits, (oldScore+scoreAdd), 0, 3 );
1870 }
1871
1872 bool ZCGame::IsDarkArea() {
1873     // is location inside, not on fire, and lights off?
1874     int playerLoc = GetNum( player.pBits, 13, 3 );
1875     int fireState = GetNum( locations[playerLoc].locBits, 4, 2 );
1876     return ( !GetBit( locations[playerLoc].locBits, (1<<7) ) && ( fireState != 2 ) )

```



```

1877 }
1878
1879 void ZCGame::LocationNotice() {
1880     int playerLoc = GetNum( player.pBits, 13, 3 );
1881     if ( IsDarkArea() ) {
1882         cout << LocationFormat( playerLoc ) << "\nThe lights are off and it is dark
here.";
1883     }
1884     else if ( !GetBit( locations[playerLoc].locBits, (1<<6) ) ) {
1885         cout << LocationDescriptionFormat( playerLoc );
1886         locations[playerLoc].locBits = SetBit( locations[playerLoc].locBits, true,
(1<<6) ); // now visited
1887     }
1888     else {
1889         cout << LocationFormat( playerLoc );
1890     }
1891 }
1892
1893 void ZCGame::FireNotice() {
1894     int playerLoc = GetNum( player.pBits, 13, 3 );
1895     int fireState = GetNum( locations[playerLoc].locBits, 4, 2 );
1896     if ( fireState == 1 )
1897         cout << "\nThe area smells of flammable fumes.";
1898     else if ( fireState == 2 ) {
1899         int fireTime = GetNum( locations[playerLoc].locBits, 0, 4 );
1900         if ( fireTime < 6 )
1901             cout << "\nA large fire has started here.";
1902         else if ( fireTime > 5 && fireTime < 13 )
1903             cout << "\nThis area is engulfed in flames and smoke.";
1904         else
1905             cout << "\nThe fire in this area is now more smoke than fire.";
1906     }
1907     else if ( fireState == 3 && !IsDarkArea() )
1908         cout << "\nThis area has been ravaged by fire, and is thoroughly burnt.";
1909 }
1910
1911 void ZCGame::ItemNotice() {
1912     if ( IsDarkArea() )
1913         return;
1914     int playerLoc = GetNum( player.pBits, 13, 3 );
1915     for (int i=0; i<12; i++) {
1916         int itemLoc = GetNum( items[i].itemBits, 2, 3 );
1917         int itemType = GetNum( items[i].itemBits, 5, 3 );
1918         if ( itemLoc == playerLoc && !GetBit( items[i].itemBits, (1<<1) ) ) {
1919             if ( !GetBit( items[i].itemBits, (1<<0) ) )
1920                 cout << "\n" << ItemFormat( itemType );
1921             else {
1922                 // used item
1923                 if ( itemType == 0 )
1924                     cout << "\nThere is a spent flare gun is here.";
1925                 else if ( itemType == 2 )
1926                     cout << "\nThere is a shattered glass bottle of alcohol is
here.";
1927                 else if ( itemType == 4 )
1928                     cout << "\nThere is the empty firing mechanism of a spear gun
is here.";
1929                 else if ( itemType == 7 )
1930                     cout << "\nThere is an empty fuel can is here.";
1931                 else
1932                     cout << "\nThere is " << ItemFormat( itemType ) << " here ...
used."; // should never be seen
1933             }
1934         }
1935     }
1936 }

```

```

1937
1938 void ZCGame::CharacterNotice() {
1939     if ( IsDarkArea() )
1940         return;
1941     int playerLoc = GetNum( player.pBits, 13, 3 );
1942     for (int i=0; i<8; i++) {
1943         int charLoc = GetNum( characters[i].charBits, 5, 3 );
1944         if ( charLoc == playerLoc ) {
1945             if ( CharacterAlive(i) ) {
1946                 // alive characters
1947                 cout << "\n" << CharacterName( i, ( GetBit( characters[i].charBits,
(1<<0) ) && !GetBit( characters[i].charBits, (1<<1) ) ) ) << " is here.";
1948             }
1949             else {
1950                 // corpses
1951                 cout << "\nThe corpse of " << CharacterName( i, ( GetBit(
characters[i].charBits, (1<<0) ) && !GetBit( characters[i].charBits, (1<<1) ) ) ) << "
lies here.";
1952             }
1953         }
1954     }
1955 }
1956
1957 void ZCGame::ZombieChatter() {
1958     int playerLoc = GetNum( player.pBits, 13, 3 );
1959     int zombieChatterCount = 0;
1960     for (int i=0; i<8; i++) {
1961         // only alive characters
1962         if ( !CharacterAlive(i) )
1963             continue;
1964         int charLoc = GetNum( characters[i].charBits, 5, 3 );
1965         // zombie, but not infected
1966         if ( charLoc == playerLoc && GetBit(characters[i].charBits, (1<<0)) && !
GetBit(characters[i].charBits, (1<<1)) ) {
1967             // chance of zombie chatter
1968             if ( rand() % 2 == 0 ) {
1969                 // additional newline before first zombie chatter
1970                 if ( zombieChatterCount == 0 )
1971                     cout << "\n";
1972                 cout << "\n " << CharacterName(i, true ) << ": " <<
CharacterDialogZombie(i);
1973                 zombieChatterCount++;
1974             }
1975         }
1976     }
1977 }
1978
1979 bool ZCGame::CharacterAlive( int charIndex ) {
1980     return ( GetNum( characters[charIndex].charBits, 2, 2 ) > 0 );
1981 }
1982
1983 void ZCGame::CharacterChatter() {
1984     int playerLoc = GetNum( player.pBits, 13, 3 );
1985     int charCount = 0;
1986     int* charactersHere = new int[8];
1987     for (int i=0; i<8; i++) {
1988         charactersHere[i] = 0;
1989         // only alive characters
1990         if ( !CharacterAlive(i) )
1991             continue;
1992         int charLoc = GetNum( characters[i].charBits, 5, 3 );
1993         if ( charLoc == playerLoc && !GetBit(characters[i].charBits, (1<<1)) && !
GetBit(characters[i].charBits, (1<<0)) ) {
1994             // only present alive non-infected non-zombie characters chatter
1995             charactersHere[charCount] = i;
1996             charCount++;

```

```

1997     }
1998 }
1999 if ( charCount > 1 ) {
2000     // chatter pick one initiate
2001     int charInitiate = charactersHere[( rand() % charCount )];
2002     cout << "\n\n " << CharacterName(charInitiate, false) << ": " <<
CharacterDialogQuestion( charInitiate );
2003     // pick a replier (not initiate)
2004     int charReplier = charInitiate;
2005     while (charReplier == charInitiate) {
2006         charReplier = charactersHere[( rand() % charCount )];
2007     }
2008     cout << "\n " << CharacterName(charReplier, false) << ": " <<
CharacterDialogAnswer( charReplier );
2009 }
2010 }
2011
2012 void ZCGame::IncrementStory() {
2013     // Present story based on timing and events
2014     bool isReleased = GetBit( gameBits, (1<<4) );
2015     bool gameOver = GetBit( gameBits, (1<<0) );
2016     int releaseCount = GetNum( gameBits, 5, 3 );
2017     int rescueCount = GetNum( scoreBits, 3, 5 );
2018
2019     if ( gameOver ) {
2020         // story end
2021     }
2022     else if ( !isReleased ) {
2023         if ( releaseCount < 7 ) {
2024             // pre-release
2025             cout << "\n" << StoryFormat(releaseCount);
2026             if ( releaseCount == 0 ) {
2027                 player.pBits = SetNum( player.pBits, 1, 13, 3 ); // on foredeck
2028                 for ( int i=0; i<8; i++ ) {
2029                     characters[i].charBits = SetNum( characters[i].charBits, 1, 5,
3 ); // on foredeck
2030                 }
2031             }
2032             else if ( releaseCount == 1 ) {
2033                 player.pBits = SetNum( player.pBits, 3, 13, 3 ); // in ballroom
2034                 for ( int i=0; i<8; i++ ) {
2035                     if ( i == 0 )
2036                         characters[i].charBits = SetNum( characters[i].charBits, 3,
5, 3 ); // capt in ballroom
2037                     else if ( i == 1 )
2038                         characters[i].charBits = SetNum( characters[i].charBits, 0,
5, 3 ); // 1st mate on bridge
2039                     else if ( i == 2 )
2040                         characters[i].charBits = SetNum( characters[i].charBits, 5,
5, 3 ); // chef in kitchen
2041                     else if ( i == 3 )
2042                         characters[i].charBits = SetNum( characters[i].charBits, 4,
5, 3 ); // phil in lounge
2043                     else
2044                         characters[i].charBits = SetNum( characters[i].charBits, 3,
5, 3 ); // rest in ballroom
2045                 }
2046             }
2047             else if ( releaseCount == 2 ) {
2048                 player.pBits = SetNum( player.pBits, 4, 13, 3 ); // in lounge
2049                 for ( int i=0; i<8; i++ ) {
2050                     if ( i < 2 )
2051                         characters[i].charBits = SetNum( characters[i].charBits, 0,
5, 3 ); // capt and 1st mate on bridge
2052                     else if ( i == 2 )
2053                         characters[i].charBits = SetNum( characters[i].charBits, 5,

```

```

5, 3 ); // chef in kitchen
2054         else
2055             characters[i].charBits = SetNum( characters[i].charBits, 4,
5, 3 ); // rest in lounge
2056     }
2057 }
2058 }
2059 }
2060 else {
2061     if ( rescueCount == 0 ) {
2062         // story response
2063     }
2064     else if ( rescueCount < 31 ) {
2065         // waiting for rescue
2066     }
2067     else {
2068         if ( rescueCount == 31 ) {
2069             if ( !GetBit( gameBits, (1<<0) ) ) {
2070                 // story rescue arrived
2071                 cout << "\n" << StoryFormat(31);
2072             }
2073             else {
2074                 cout << "\n" << StoryFormat(32);
2075             }
2076         }
2077     }
2078 }
2079 }
2080
2081 void ZCGame::IncrementInfection() {
2082     // Three-stages : infection but not zombie, both infection and zombie, then
zombie but not infected
2083     for ( int i=0; i<8; i++ ) {
2084         if ( !CharacterAlive(i) )
2085             continue; // only alive characters
2086         if ( !GetBit( characters[i].charBits, (1<<0) ) && GetBit( characters[i].
charBits, (1<<1) ) ) {
2087             // any character not zombie but infected becomes zombie
2088             if ( GetNum( player.pBits, 13, 3 ) == GetNum( characters[i].charBits, 5
, 3 ) ) {
2089                 // if in same room as player, perform resting dialog
2090                 cout << "\n " << CharacterName( i, false ) << ": " <<
CharacterDialogResting( i );
2091             }
2092             characters[i].charBits = SetBit( characters[i].charBits, true, (1<<0)
); // zombie, starting next turn
2093             characters[i].charBits = SetBit( characters[i].charBits, true, (1<<4)
); // stay put
2094         }
2095         else if ( GetBit( characters[i].charBits, (1<<0) ) && GetBit( characters[i
].charBits, (1<<1) ) ) {
2096             // character turns
2097             if ( GetNum( player.pBits, 13, 3 ) == GetNum( characters[i].charBits, 5
, 3 ) ) {
2098                 // if in same room as player, announce character turn
2099                 cout << "\n" << CharacterName( i, false ) << " has turned zombie.";
2100             }
2101             characters[i].charBits = SetBit( characters[i].charBits, false, (1<<1)
); // no longer 'infected', but full zombie
2102             characters[i].charBits = SetBit( characters[i].charBits, true, (1<<4)
); // stay put
2103         }
2104     }
2105 }
2106
2107 void ZCGame::IncrementFire() {

```

```

2108 // any area on fire, increment fire timer
2109 // if timer at end, fire out and room set to 'burnt'
2110 // if timer more than 7, chance of spread through exit to another room, if
flammable
2111 for ( int i=0; i<8; i++ ) {
2112     if ( GetNum( locations[i].locBits, 4, 2 ) == 2 ) {
2113         // room on fire
2114         int fireTimer = GetNum( locations[i].locBits, 0, 4 );
2115         fireTimer++;
2116         if ( fireTimer >= 15 ) {
2117             fireTimer = 15;
2118             locations[i].locBits = SetNum( locations[i].locBits, 3, 4, 2 ); //
area 'burnt'
2119         }
2120         locations[i].locBits = SetNum( locations[i].locBits, fireTimer, 0, 4 );
2121         int chanceOfSpread = ( rand() % 4 );
2122         if ( fireTimer < 15 && fireTimer > 6 && chanceOfSpread == 0 ) {
2123             int spreadFire = ( rand() % 3 );
2124             int fireExit = FindRoomFromExit( i, spreadFire );
2125             if ( GetNum( locations[fireExit].locBits, 4, 2 ) == 1 ) {
2126                 // fire spread to adjacent flammable room
2127                 locations[fireExit].locBits = SetNum( locations[fireExit].
locBits, 2, 4, 2 ); // on fire
2128                 locations[fireExit].locBits = SetNum( locations[fireExit].
locBits, 1, 0, 4 ); // fire timer reset
2129                 // if player in room, announce
2130                 if ( GetNum( player.pBits, 13, 3 ) == fireExit )
2131                     cout << "\nFire has spread into this room\n";
2132             }
2133         }
2134     }
2135 }
2136 }
2137
2138 void ZCGame::FireDamage() {
2139     for (int i=0; i<8; i++) {
2140         int fireTime = GetNum( locations[i].locBits, 0, 4 );
2141         if ( GetNum( locations[i].locBits, 4, 2 ) == 2 && ( fireTime > 5 &&
fireTime < 13 ) ) {
2142             // fire damage knocks lights out
2143             if ( fireTime > 8 ) {
2144                 if ( (rand() % 4) == 0 ) {
2145                     locations[i].locBits = SetBit( locations[i].locBits, false, (1
<<7) );
2146                 }
2147             }
2148             // player hurt by fire
2149             if ( GetNum( player.pBits, 13, 3 ) == i ) {
2150                 int playerHealth = GetNum( player.pBits, 11, 2 );
2151                 playerHealth--;
2152                 if ( playerHealth <= 0 ) {
2153                     playerHealth = 0;
2154                     cout << "\n\n - You have died in the flames -";
2155                     gameBits = SetBit( gameBits, true, (1<<0) ); // game over
2156                 }
2157                 else {
2158                     cout << "\n\nYou have been burned by the flames.";
2159                 }
2160                 player.pBits = SetNum( player.pBits, playerHealth, 11, 2 );
2161             }
2162             // characters and zombies hurt by fire
2163             for (int n=0; n<8; n++) {
2164                 if ( GetNum( characters[n].charBits, 5, 3 ) == i ) {
2165                     if ( CharacterAlive(n) ) {
2166                         // character is here and not dead
2167                         int charHealth = GetNum( characters[n].charBits, 2, 2 );

```

```

2168         charHealth--;
2169         if ( charHealth <= 0 ) {
2170             charHealth = 0;
2171             // if player in room, announce char death
2172             if ( GetNum(player.pBits, 13, 3) == i )
2173                 cout << "\n" << CharacterName( n, ( !GetBit(
characters[n].charBits, (1<<1)) ) ) << " is killed by fire.";
2174         }
2175         else {
2176             // if non-zombie and if player in room, char pain
dialog
2177             if ( GetNum(player.pBits, 13, 3) == i && ( GetBit(
characters[n].charBits, (1<<1)) || !GetBit( characters[n].charBits, (1<<0)) ) )
2178                 cout << "\n " << CharacterName( n, ( !GetBit(
characters[n].charBits, (1<<1)) && GetBit(characters[n].charBits, (1<<0)) ) ) << ": "
<< CharacterDialogExclamation( n );
2179         }
2180         characters[n].charBits = SetNum( characters[n].charBits,
charHealth, 2, 2 );
2181     }
2182 }
2183 }
2184 }
2185 }
2186 }
2187
2188 void ZCGame::DoZombieMoves() {
2189     for (int i=0; i<8; i++) {
2190         // only alive characters
2191         if ( !CharacterAlive(i) )
2192             continue;
2193         // only zombies
2194         // zombie, but not infected
2195         if ( GetBit( characters[i].charBits, (1<<0)) && !GetBit( characters[i].
charBits, (1<<1)) ) {
2196             int currRoom = GetNum( characters[i].charBits, 5, 3 );
2197             int newRoom = FindRoomFromExit( currRoom, ( rand() % 3 ) );
2198             // interrupt move if player is in same room
2199             // do move now if target room is not same as current (zombies move
slow)
2200             // set current room to target and return
2201             if ( !GetBit( characters[i].charBits, (1<<1)) && currRoom != GetNum(
player.pBits, 13, 3 ) ) {
2202                 // notify player in same room that zombie is entering
2203                 if ( newRoom == GetNum( player.pBits, 13, 3 ) && !IsDarkArea() )
2204                     cout << "\n\n" << CharacterName(i, true) << " arrives here.\n";
2205                 characters[i].charBits = SetNum( characters[i].charBits, newRoom, 5
, 3 );
2206                 return;
2207             }
2208             bool charsInRoom = false;
2209             for (int n=0; n<8; n++) {
2210                 if ( n != i && CharacterAlive(n) && currRoom == GetNum( characters[
n].charBits, 5, 3 ) && !GetBit( characters[n].charBits, (1<<0)) && !GetBit( characters[n
].charBits, (1<<1)) ) {
2211                     charsInRoom == true;
2212                     break;
2213                 }
2214             }
2215             // if non-zombie / non-infected in room, stay (target room is same)
2216             // if player in room, stay
2217             // if no non-zombie in room, chance of move (un-set 'waiting' bit)
2218             if ( charsInRoom || currRoom == GetNum( player.pBits, 13, 3 ) ) {
2219                 characters[i].charBits = SetBit( characters[i].charBits, true, (1<<
4) );
2220             }

```

```

2221         else {
2222             characters[i].charBits = SetBit( characters[i].charBits, false, (1
2223 <<4) );
2224         }
2225     }
2226 }
2227
2228 void ZCGame::DoZombieAttacks() {
2229     for (int i=0; i<8; i++) {
2230         // only alive characters
2231         if ( !CharacterAlive(i) )
2232             continue;
2233         // only zombies
2234         // zombie, but not infected
2235         if ( GetBit( characters[i].charBits, (1<<0) ) && !GetBit( characters[i].
charBits, (1<<1) ) ) {
2236             int currRoom = GetNum( characters[i].charBits, 5, 3 );
2237             int* allChars = new int[8];
2238             int charsInRoom = 0;
2239             for (int n=0; n<8; n++) {
2240                 allChars[charsInRoom] = 0;
2241                 // only alive characters
2242                 if ( !CharacterAlive(n) )
2243                     continue;
2244                 if ( n != i && currRoom == GetNum( characters[n].charBits, 5, 3 )
&& !GetBit( characters[n].charBits, (1<<0) ) && !GetBit( characters[n].charBits, (1<<1) )
) {
2245                     allChars[charsInRoom] = n;
2246                     charsInRoom++;
2247                 }
2248             }
2249             // if non-zombie / non-infected in room, attack
2250             // if player in room, and not dead, attack
2251             if ( charsInRoom > 0 || ( currRoom == GetNum( player.pBits, 13, 3 ) &&
GetNum(player.pBits, 11, 2 ) > 0 ) ) {
2252                 if ( currRoom == GetNum( player.pBits, 13, 3 ) && GetNum(player.
pBits, 11, 2 ) > 0 ) {
2253                     // zombie attack player
2254                     cout << "\n" << CharacterName( i, true ) << " attacks you";
2255                     // if nothing equipped, 75% chance lose health
2256                     // if one item equipped, 50% chance lose health
2257                     // if two items equipped, 25% chance lose health
2258                     bool loseHealth = true;
2259                     int combatRoll = ( rand() % 4 );
2260                     if ( combatRoll == 0 )
2261                         loseHealth = false;
2262                     if ( GetNum(player.pBits, 7, 4 ) != 15 && combatRoll < 2 )
2263                         loseHealth = false;
2264                     if ( GetNum(player.pBits, 3, 4 ) != 15 && combatRoll < 3 )
2265                         loseHealth = false;
2266                     if ( loseHealth ) {
2267                         int pHealth = GetNum( player.pBits, 11, 2 );
2268                         pHealth -= 1;
2269                         player.pBits = SetNum( player.pBits, pHealth, 11, 2 );
2270                         if ( pHealth < 1 ) {
2271                             // player dies
2272                             cout << "\n\n - YOU HAVE TURNED ZOMBIE -";
2273                             gameBits = SetBit( gameBits, true, (1<<0) );
2274                         }
2275                     }
2276                     else {
2277                         cout << "\n - YOU ARE HURT -";
2278                     }
2279                 }
2280                 else {
2281                     cout << " and misses.";

```

```

2281     }
2282 }
2283 else {
2284     // infect random character
2285     int infectedChar = allChars[( rand() % charsInRoom )];
2286     characters[infectedChar].charBits = SetBit( characters[
infectedChar].charBits, true, (1<<1) );
2287     // reduce character health by one (set to 2)
2288     characters[infectedChar].charBits = SetNum( characters[
infectedChar].charBits, 2, 2, 3 );
2289     // character performs exclamation
2290     if ( currRoom == GetNum( player.pBits, 13, 3 ) )
2291         cout << "\n " << CharacterName( infectedChar, false ) << ":
" << CharacterDialogExclamation( infectedChar );
2292     }
2293 }
2294 }
2295 }
2296 }
2297
2298 void ZCGame::DoCharacterMoves() {
2299     bool choseToExit = ( rand() % 6 == 0 );
2300     int chosenExitIndex = ( rand() % 3 );
2301     for (int i=0; i<8; i++) {
2302         // only alive characters
2303         if ( !CharacterAlive(i) )
2304             continue;
2305         // only non-zombies, set target room
2306         if ( !GetBit( characters[i].charBits, (1<<0) ) ) {
2307             // character moves
2308             int currRoom = GetNum( characters[i].charBits, 5, 3 );
2309             int targetRoom = currRoom;
2310             int chosenExit = FindRoomFromExit( currRoom, chosenExitIndex );
2311             // if infected, stay
2312             if ( GetBit( characters[i].charBits, (1<<1) ) ) {
2313                 characters[i].charBits = SetBit( characters[i].charBits, true, (1<<
4) );
2314             }
2315             else {
2316                 bool zombieInRoom = false;
2317                 bool otherInRoom = false;
2318                 for (int n=0; n<8; n++) {
2319                     // only alive, and not self
2320                     if ( n != i && CharacterAlive(n) && GetNum( characters[n].
charBits, 5, 3 ) == currRoom ) {
2321                         // zombie, but not infected
2322                         if ( GetBit( characters[n].charBits, (1<<0) ) && !GetBit(
characters[n].charBits, (1<<1) ) ) {
2323                             zombieInRoom = true;
2324                         }
2325                         else
2326                             otherInRoom = true;
2327                     }
2328                 }
2329                 if ( GetNum( locations[currRoom].locBits, 4, 2 ) == 2 ) {
2330                     // if current room on fire, move
2331                     int panicExit = FindRoomFromExit( currRoom, ( rand() % 3 ) );
2332                     targetRoom = panicExit;
2333                 }
2334                 else if ( zombieInRoom ) {
2335                     // if zombie in room, move
2336                     int panicExit = FindRoomFromExit( currRoom, ( rand() % 3 ) );
2337                     targetRoom = panicExit;
2338                 }
2339                 else if ( GetNum( player.pBits, 13, 3 ) == currRoom ) {
2340                     // if player in room, stay

```



```

2341         characters[i].charBits = SetBit( characters[i].charBits, true,
(1<<4) );
2342     }
2343     else if ( !otherInRoom ) {
2344         // if alone, chance of move through any exit
2345         if ( rand() % 4 == 0 ) {
2346             int panicExit = FindRoomFromExit( currRoom, ( rand() % 3 )
);
2347             targetRoom = panicExit;
2348         }
2349     }
2350     else {
2351         if ( ( rand() % 8 ) == 0 ) {
2352             // if not alone, very rare chance of split up
2353             int panicExit = FindRoomFromExit( currRoom, ( rand() % 3 )
);
2354             targetRoom = panicExit;
2355         }
2356         else if ( choseToExit ) {
2357             // if not alone, rare chance of move through same exit
2358             targetRoom = chosenExit;
2359         }
2360     }
2361 }
2362 // do move now (characters move fast)
2363 if ( targetRoom != currRoom ) {
2364     if ( currRoom == GetNum( player.pBits, 13, 3 ) && !IsDarkArea() )
2365         cout << "\n" << CharacterName( i, false ) << " leaves.";
2366     // if target room != current room, set current to target
2367     characters[i].charBits = SetNum( characters[i].charBits, targetRoom
, 5, 3 );
2368     if ( targetRoom == GetNum( player.pBits, 13, 3 ) && !IsDarkArea() )
2369         cout << "\n" << CharacterName( i, false ) << " arrives here.";
2370 }
2371 }
2372 }
2373 }
2374
2375 void ZCGame::DoPlayerAttack() {
2376     // find all zombies in same room, pick one
2377     int numZombies = 0;
2378     int* zombiesInRoom = new int[8];
2379     for (int i=0; i<8; i++) {
2380         zombiesInRoom[numZombies] = 0;
2381         // only alive characters that are in room
2382         if ( !CharacterAlive(i) || GetNum( characters[i].charBits, 5, 3 ) != GetNum
( player.pBits, 13, 3 ) )
2383             continue;
2384         // zombie, but not infected
2385         if ( GetBit( characters[i].charBits, (1<<0) ) && !GetBit( characters[i].
charBits, (1<<1) ) ) {
2386             zombiesInRoom[numZombies] = i;
2387             numZombies++;
2388         }
2389     }
2390     if ( numZombies == 0 ) {
2391         cout << "\nThere is no threat to attack here.";
2392     }
2393     else {
2394         int targetZombie = zombiesInRoom[( rand() % numZombies )];
2395         int combatRoll = ( rand() % 4 );
2396         int damageDone = 0;
2397         // if equipped, use either equipped item as weapon
2398         int equipA = GetNum( player.pBits, 7, 4 );
2399         int equipB = GetNum( player.pBits, 3, 4 );
2400         int weaponIndex = 15;

```

```

2401     if ( equipA != 15 && equipB != 15 ) {
2402         if ( rand() % 2 == 0 )
2403             weaponIndex = equipA;
2404         else
2405             weaponIndex = equipB;
2406     }
2407     else if ( equipA != 15 )
2408         weaponIndex = equipA;
2409     else if ( equipB != 15 )
2410         weaponIndex = equipB;
2411     // otherwise, no damage
2412     if ( weaponIndex != 15 ) {
2413         // find item weapon type
2414         int weaponType = GetNum( items[weaponIndex].itemBits, 5, 3 );
2415         switch (weaponType) {
2416             case 0:
2417                 damageDone = 3;
2418                 break;
2419             case 1:
2420                 damageDone = 2;
2421                 break;
2422             case 2:
2423                 damageDone = 1;
2424                 break;
2425             case 3:
2426                 damageDone = 2;
2427                 break;
2428             case 4:
2429                 damageDone = 3;
2430                 break;
2431             case 5:
2432                 damageDone = 2;
2433                 break;
2434             case 6:
2435                 damageDone = 2;
2436                 break;
2437             case 7:
2438                 damageDone = 1;
2439                 break;
2440         }
2441         if ( combatRoll < 3 && damageDone > 0 ) {
2442             // hit, reduce zombie health
2443             int zombieHealth = GetNum( characters[targetZombie].charBits, 2, 2
);
2444             zombieHealth -= damageDone;
2445             if ( zombieHealth <= 0 ) {
2446                 zombieHealth = 0;
2447                 // zombie killed
2448                 cout << "\n - " << CharacterName( targetZombie, true ) << " is
killed with " << ItemName(weaponType) << " -";
2449                 // GAME BALANCE NEED
2450                 int numZombiesOnboard = 0;
2451                 int numCharsAlive = 0;
2452                 for ( int i=0; i<8; i++ ) {
2453                     if ( GetBit( characters[i].charBits, (1<<0) ) ) {
2454                         numZombiesOnboard++;
2455                     }
2456                     else {
2457                         numCharsAlive++;
2458                     }
2459                     if ( numCharsAlive > 0 && numZombiesOnboard == 0 ) {
2460                         gameBits = SetBit( gameBits, false, (1<<4) ); // reset
infection release timer
2461                     }
2462                     else {
2463                         // no chars left to infect, player is alone onboard

```

```

2464     }
2465     }
2466     }
2467     else
2468         cout << "\nYou hit " << CharacterName( targetZombie, true ) <<
" with " << ItemName(weaponType) << " for " << damageDone << " damage.";
2469         characters[targetZombie].charBits = SetNum( characters[targetZombie
].charBits, zombieHealth, 2, 2 );
2470         // use item
2471         if ( UseItem( weaponIndex ) ) {
2472             if ( weaponIndex != 1 ) {
2473                 // one-use, so drop item
2474                 DropItem( weaponIndex );
2475                 cout << "\n";
2476             }
2477         }
2478     }
2479     else {
2480         cout << "\nYour attempt to hit " << CharacterName( targetZombie,
true ) << " missed.";
2481         if ( weaponIndex == 15 )
2482             cout << "\n[HINT] Equip yourself with items you take to be more
effective in combat.\n";
2483         else
2484             cout << "\n";
2485     }
2486 }
2487 else {
2488     cout << "\nYour unarmed attack is ineffective against the undead.";
2489     cout << "\n[HINT] Equip yourself with items you take to be more
effective in combat.\n";
2490 }
2491 }
2492 }
2493
2494 void Pause( int seconds ) {
2495     int pTime = time(NULL);
2496     while (time(NULL)<(pTime+seconds)) {
2497         // ... waiting
2498     }
2499 }
2500
2501 int main() {
2502
2503     // PROTOTYPE TESTING
2504     ZCGame thisGame;
2505     // SEED RANDOM
2506     srand(time(NULL));
2507     // INITIALIZE GAME
2508     thisGame.Initialize();
2509
2510     thisGame.player.pBits = SetNum( thisGame.player.pBits, 4, 13, 3 );
2511     thisGame.player.pBits = SetNum( thisGame.player.pBits, 3, 11, 2 ); //
location=4, health=3
2512     thisGame.player.pBits = SetNum( thisGame.player.pBits, 15, 7, 4 ); // equip
Item A = none
2513     thisGame.player.pBits = SetNum( thisGame.player.pBits, 15, 3, 4 ); // equip
Item B = none
2514
2515     cout << "\n [ ZOMBIE CRUISE . a text adventure by Glenn Storm ]\n\n";
2516
2517     Pause(3);
2518
2519     cout << HelpFormat();
2520
2521     Pause(4);

```

```

2522
2523     string testStr = "debug";
2524
2525     while ( testStr != "quit" && !GetBit( thisGame.gameBits, (1<<0) ) ) {
2526         // story
2527         thisGame.IncrementStory();
2528         // player view
2529         thisGame.LocationNotice();
2530         thisGame.FireNotice();
2531         thisGame.IncrementFire();
2532         thisGame.ItemNotice();
2533         thisGame.CharacterNotice();
2534         // chatter
2535         thisGame.ZombieChatter();
2536         thisGame.CharacterChatter();
2537         // zombie attacks
2538         thisGame.DoZombieAttacks();
2539         // zombie moves
2540         thisGame.DoZombieMoves();
2541         // infection progress
2542         thisGame.IncrementInfection();
2543         // character moves
2544         thisGame.DoCharacterMoves();
2545         // fire damage
2546         thisGame.FireDamage();
2547         cout << "\n\n";
2548         Pause(1);
2549         if ( !GetBit( thisGame.gameBits, (1<<0) ) ) { // game not over
2550             // player input
2551             testStr = PlayerPrompt();
2552             thisGame.ParseMove(testStr);
2553             cout << "\n";
2554         }
2555         else {
2556             cout << " . GAME OVER .\n";
2557             Pause(3);
2558         }
2559     }
2560
2561     // GAME LOOP
2562
2563     // Game update
2564     // Story update
2565     // Fire update
2566     // Character update
2567     // Zombie update
2568     // Player update
2569     // Score update
2570
2571     // Story format
2572     // Location format
2573     // Dynamics (lights/fire) format
2574     // Item format
2575     // Character format
2576     // Zombie format
2577
2578     // Dialog format
2579
2580     // Present story
2581     // Present location
2582     // Present dynamics
2583     // Present items
2584     // Present characters
2585     // Present zombies
2586     // Present dialog
2587

```

```

2588 // Player prompt
2589
2590 // Game reaction (quit)
2591 // Help reaction
2592 // Move reaction
2593 // Talk reaction
2594 // Look reaction
2595 // Use reaction
2596 // Equip reaction
2597
2598 // Inventory management
2599
2600 // Combat response
2601 // Item response
2602 // Character response
2603 // Zombie response
2604 // Fire response
2605 // Lighting response
2606
2607 // Radio response
2608 // Location-specific response
2609
2610 // END GAME LOOP
2611
2612 // TEMP scoring
2613 int numChars = 0;
2614 int numZombies = 0;
2615 for (int i=0; i<8; i++) {
2616     // only alive characters
2617     if ( !thisGame.CharacterAlive(i) )
2618         continue;
2619     if ( GetBit( thisGame.characters[i].charBits, (1<<0) ) )
2620         numZombies++;
2621     else
2622         numChars++;
2623 }
2624 if ( GetNum( thisGame.player.pBits, 11, 2 ) < 1 )
2625     numZombies++; // player is zombie if dead
2626 else
2627     numChars++; // player is surviving character if alive
2628 cout << "\n\n TOTAL ZOMBIES ONBOARD: " << numZombies;
2629 cout << "\n TOTAL SURVIVORS ONBOARD: " << numChars;
2630 int score = 0;
2631 if ( GetBit( thisGame.gameBits, (1<<2) ) ) {
2632     score += 1; // SOS called +1
2633 }
2634 if ( GetBit( thisGame.gameBits, (1<<1) ) ) {
2635     score += 1; // Rescue arrived +1
2636     if ( GetNum( thisGame.player.pBits, 11, 2 ) > 0 ) {
2637         score += 1; // Player alive
2638     }
2639     if ( numChars > 3 ) {
2640         score += 1; // Characters outnumber zombies
2641     }
2642     if ( numZombies < 4 ) {
2643         score += 3; // Zombies suppressed
2644     }
2645 }
2646 cout << "\n\n FINAL SCORE: " << score << "/7\n" << " RANK: ";
2647 // Score Ranks: <3 "Zombie Meat", 3-4 "Survivor", 5-6 "Hero", 7 "Zombie Killer"
2648 if ( score < 3 )
2649     cout << "Zombie Meat";
2650 else if ( score < 5 )
2651     cout << "Survivor";
2652 else if ( score < 7 )
2653     cout << "Hero";

```

```
2654     else
2655         cout << "Zombie Killer";
2656     cout << "\n";
2657
2658     Pause(5);
2659
2660     string e;
2661     cout << "\n [PRESS ENTER KEY TO END]\n";
2662     getline(cin, e);
2663
2664     return 0;
2665 };
```