Name

In the early 26th century the 50XFTL drive was developed and the stars became open to mankind. Warping space and propelling starships 50 times faster than light mankind left their crowded solar system for the frontier of space. By the mid 29th century giant space stations called starports existed at 10 of the most densely populated star systems. These starports provided fuel and services for starships as well as acting as a trade center for the solar system. Each starport was given a classification which has less to do with what would be available and more to do with the prices you can generally expect. Unexpected events can cause unpredictable price changes leading to unexpected profit for you.

Star Merchant is a futuristic trading simulation. As you travel to star systems pay attention to the percentage column (%). This shows what percentage of the market rate the price is set at for that particular item at that starport at that instance, 100% being the going market rate for that particular cargo as set by the Interstellar Trade Commission. The 100% rate for each cargo is defined in cargo.h. Using this can help you buy low and sell high, which is the name of the game.

When you travel be sure to keep some money in your account for expenses like docking fees, fuel, and your crew salary. If you can't af-

/* upgrades.h linsting begins: */

```
#ifndef __UPGRADES_H
#define __UPGRADES_H

#define MAX_UPGRADES 7

typedef struct sShipUpgrade {
    const char *package_name;
    float engine_speed;
    long upgrade_cost;
    short tonnage;
} SHIPUPGRADE;

SHIPUPGRADE upgrades[MAX_UPGRADES] = {
    {"50XFTL",0.02,500000,200},
    {"67XFTL",0.015,1500000,210},
    {"Einsteinium Drive Mk. I",0.01,3750000,220},
    {"Einsteinium Drive Mk. II",0.00525,65000000, 250},
    {"Terradrive Mk. I",0.005,100000000,300},
    {"Terradrive Mk. II",0.0045,275000000,400},
    {"Infinidrive",0.003,500000000,600}
};

#endif
```

/* starports.h linsting begins: */

```
#ifndef __STARPORTS_H
#define __STARPORTS_H

#define MAX_STARPORTS 11

typedef struct sStarport {
    const char *port_name;
    const char *classification;
    float X1;
    float Y1;
    float distance;  /* This is calculated and then stored here with each "jump"
*/
    int direction;   /* same for this one as well */
    int trade;
    const char *spectral_type;
} STARPORT;

STARPORT starports[MAX_STARPORTS] = {
    {"Lalande 21185","NI,P", 2.83, -7.36, 0.00, 0, 20, "M2V"},
    {"Alpha Centauri","NA,I", -2.51, -3.57, 0.00, 0, 9, "G2V"},
    {"Sirius","A", 8.38, 9.93, 0.00, 0, 2, "A1V"},
    {"Barnards Star","I,P", -6.1, 0, 0.00, 0, 24, "M4V"},
    {"Sol","R",0, 0, 0.00, 0, 32, "G2V"},
    {"Ross 154","NI,NA", -8.87, 2.05, 0.00, 0, 5, "M3.5V"},
    {"Epsilon Eridani","A,P", 8.45, 6.65, 0.00, 0, 18, "K2V"},
    {"Luyten 726-8","NA", 2.99, 7.42, 0.00, 0, 1, "M6V"},
    {"Luyten 789-6","A,NI,P", -4.43, 9.3, 0.00, 0, 22, "M5V"},
    {"Ross 248","A,I", -0.89, 10.26, 0.00, 0, 10, "M5.5V"},
    {"Gliese 876","R,I", 5.38, -14.33, 0.00, 0, 25, "M3.5V"}
};

int trade_factors[6] = {0,0,0,0,0,0};

#endif
```

/* Listing continued on next page…*/

/* cargo.h listing begins */

```
#ifndef __CARGO_H
#define __CARGO_H

#define MAX_CARGO_PARTITIONS 20
#define MAX_CARGO_FOR_SALE 8
#define MAX_CARGOS 38
#define MAX_TONS 200

#ifndef FALSE
       #define FALSE 0
#endif

typedef struct sCargoPartition {
    int cargo_type;
    int amt;
    long price;
    int pct;
    int purchase_loc;
    int in_port;
} CARGOPARTITION;


CARGOPARTITION cargo_bays[MAX_CARGO_PARTITIONS] = {
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE}
};

CARGOPARTITION starport_cargo[MAX_CARGO_FOR_SALE] = {
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE},
    {-1,0,0,0,0,FALSE}
};

typedef struct sCargoType {
    const char *cargo_name;
    int price_factors[6];
    long long basePrice;
    short amt_multiplier;
} CARGOTYPE;
```

/* Listing continued on next page…*/

ford your expenses upon your arrival your inability to afford the docking fees will cause your ship to be confiscated, your crew to be let go, and your cargo sold to cover the exorbitant cost of your legal fees and you will spend the rest of your life stuck on the starport flipping soy patties. At the end of 2 years the lease on your ship comes due. If you don't have 2 million in the bank by then they'll repossess your starship, too.

Star Merchant has multiple source files that will all need to be included, so be sure to pay attention to the file names as you're typing.

When the original game was published there were described a number of features that were not implemented in the final game. This version stays true to the original game, but it does leave a number of possibilities for expansion. For instance:

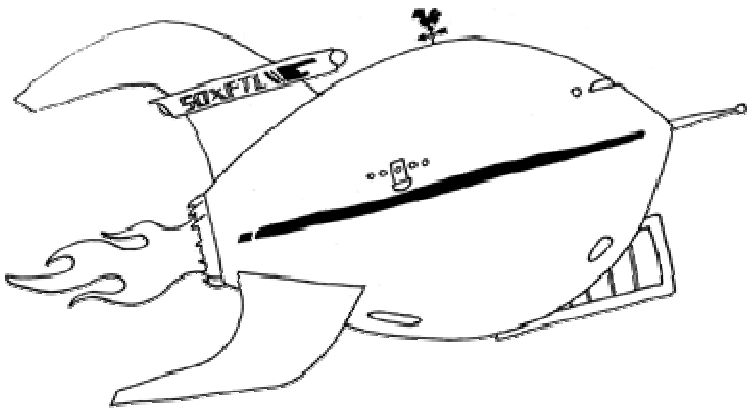
- At random space pirates will warp directly into your cargo holds and steal 10% of your cargo.
- The mass that you're carrying should affect the time to destination, thus affecting the cost of traveling.
- Expand the upgrade system to be more piecemeal so you can upgrade the engines but downgrade your cargo holds (preferable later in the game).

There are no end of other modification that can be made to this game with just a little imagination.

Star Merchant was written by Devin Watson, inspired by a BASIC game by the same name by Lloyd Johnson as found in *Big Computer Games* edited by David H. Ahl © 1984.

/* Listing continued from previous page */

```
CARGOTYPE cargos[MAX_CARGOS] = {
    {"Crystals", {3, -2, 2, -2, 0, -4}, 20000, 1},
    {"Radioactives", {0, 1, 4, -3, 0, -2},1000000, 1},
    {"Alloys", {-2, 0, -4, 6, 1, -2}, 200000, 1},
    {"Medicine", {-1, 4, -4, 3, -2, 0},100000, 1},
    {"Gems", {4, -2, 4, -4, -1, 1}, 1000000, 1},
    {"Aircraft", {-2, 4, -3, 4, 1, -1}, 1000000, 1},
    {"Grav Sleds", {2, 0, -1, 1, 0, 0}, 6000000, 1},
    {"Neurocomputers", {1, 0 ,-2, 0, 0, 0},  10000000, 1},
    {"Terraformers", {1, 0 ,-1, 0, 0, 0},  25000000, 1},
    {"Planetary Matrix", {1, 0, -1, 0, 0}, 75500000, 1},
    {"ATV", {-2, 2, -2, 1, 1, 0}, 300000, 1},
    {"AFV", {0, 2, -2, 0, 0, 1}, 700000, 1},
    {"Firearms", {-2, 6, -4, 1, 0, 0}, 30000, 2},
    {"Ammunition", {-1, 6, -5, 2, 0, 0}, 30000, 2},
    {"Plasma Guns", {-1, 3, -2, 0, 0, 0}, 200000, 2},
    {"Tools", {-4, 7, -8, 4, 5, 0}, 10000, 2},
    {"Body Armor", {-3, 6, -4, 1, 0, 0}, 50000, 2},
    {"Farm Machinery", {-2, 2, -6, 0, 6, -4}, 150000, 1},
    {"Liquor", {3, 3, -1, 0, -3, 0}, 10000, 1},
    {"Silver", {3, -1, 3, -1, 0, -2}, 70000, 1},
    {"Spices", {4, -2, 3, -1, -5, 2}, 6000, 1},
    {"Electronics", {0, 0, -4, 4, 1, 1}, 100000, 1},
    {"Mechanical Parts", {0, 1, -3, 3, 2, 1}, 75000, 1},
    {"Cybernetic Parts", {1, 0, -4, 2, 1, 0}, 250000, 1},
    {"Computer Parts", {-1, 0, -2, 1, 0}, 250000, 1},
    {"Machine Tools", {1, 0, -2, 1, 0, 0}, 750000, 1},
    {"Space Suits", {-1, 2, -3, 2, 2, 0}, 400000, 1},
    {"Fruit", {1, 2, 3, 3, -4, -6}, 1000, 2},
    {"Textiles", {3, 0, -3, 1, -4, -3}, 3000, 3},
    {"Polymers", {-2, 0 ,3, 3, 0, 0}, 7000, 4},
    {"Meat", {0, 0, 5, 2, -5, 5}, 1500, 4},
    {"Petrochemicals", {2, 0, 4, -2, 3, 0}, 10000, 6},
    {"Grain", {0, 0, 1, 3, -5, 6}, 300, 8},
    {"Wood", {0, 0, 1, 2, -7, 3}, 1000, 2},
    {"Copper", {2, 2, 3, -2, -1, -3}, 2000, 2},
    {"Tin", {2, 2, 3, -4, -1, -2}, 9000, 3},
    {"Steel", {-1, 2, 6, 0, 0, 0}, 500, 4},
    {"Aluminum", {-1, 1, 3, -2, 0, -2}, 1000, 5}
};

#define PRICE_DATA_MAX 14
float price_data[PRICE_DATA_MAX] = {
    0.400,
    0.500,
    0.700,
    0.800,
    0.900,
    1.000,
    1.100,
    1.200,
    1.300,
    1.500,
    1.700,
    2.000,
    3.000,
    4.000
};

#endif
```



/* Listing continued on next page…*/

/* main.c listing begins */

```
#ifndef _XOPEN_SOURCE_EXTENDED
# define _XOPEN_SOURCE_EXTENDED 1
#endif

#include <ctype.h>
#include <string.h>
#include <curses.h>
#include <time.h>
#include <ctype.h>
#include <stdlib.h>
#include <locale.h>
#include <math.h>
#include <limits.h>
#include "cargo.h"
#include "starports.h"
#include "upgrades.h"

#ifdef WACS_S1
# define HAVE_WIDE 1
#else
# define HAVE_WIDE 0
#endif

#if HAVE_WIDE
# include <wchar.h>
#endif

#if defined(PDCURSES) && !defined(XCURSES)
# define HAVE_RESIZE 1
#else
# define HAVE_RESIZE 0
#endif

#ifdef A_COLOR
# define HAVE_COLOR 1
#else
# define HAVE_COLOR 0
#endif

#ifdef ACS_S3
# define ACSNUM 32
#else
# define ACSNUM 25
#endif

/* Game states */
#define GAME_INTRO 1
#define GAME_PLAYING 2
#define GAME_DONE 3

#define SQR(X) ((X) * (X))

/* Game variables */
int game_state;
int width;
int height;
long bankAccount;
float shipTime;
float elapsedTime;
long crew_pay_rate;
int cur_starport;                   /* Current starport */
short partitions_available;         /* Amount of available cargo space on ship */
short game_seed;
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
int port_cargos_remaining;
int num_runs = 0;
int ship_tonnage;
short ship_damage;
short ship_level;
float totalDistance;

/* Game constants */
#define DOCKING_FEE 50000         /* 50,000 credits for docking at starport */
#define FUEL_COST 100000          /* 100,000 credits per lightyear of fuel */
#define CREW_STARTING_PAY 500000  /* 500,000 credits per ship year for crew */
#define LEASE_COST 2000000        /* Lease is 2,000,000 credits for 2 years */
#define DAMAGE_SLOWDOWN 0.01
#define C_MS 299792458            /* Speed of light (c), in m/s, used for engine
speed calculations */

#ifndef M_PI
       #define M_PI 3.14159265358979323846264338
#endif

#define RED_PAIR 4
#define CYAN_PAIR 3

WINDOW *win;        /* Main game area */
WINDOW *hud;        /* Heads-up display */
WINDOW *manifest;   /* Ship's manifest of cargo */
WINDOW *infowin;    /* Information popup window */
WINDOW *shipstatus; /* Status display of ship condition */
bool quit = FALSE;

char *intro_msg[2] = {
  "You have just spent 2 million credits on a 2-year lease for a merchant",
  "starship.  This leaves you with 2 million credits operating capital."};
char *intro_msg2[3] = {
  "Your ship can hold a total of 20 cargos with a total cargo weight of 200",
  "tons. The fuel capacity of your ship is great enough such that travel ",
  "between any 2 starports is possible without refueling."};
char *intro_msg3[2] = {
  "You are presently traveling from Alpha Centauri to Sol.  You are",
  "carrying no cargo."};
char *intro_msg4[2] = {
  "The goal of the game is simple: survive to make it to 1 billion ",
  "credits so you can retire on your own personal planet."};

char *intro_msg5[2] = {
  "The Starport Trade Classification determines the cargo price but does not",
  "determine which cargos are available."};
char *intro_msg6[1] = {"Abbreviations used for trade class are as follows:"};
char *intro_msg7[6] = {" R: Rich"," P: Poor", " I: Industrial",
  "NI: Non-industrial", " A: Agricultural","NA: Non-agricultural"};

void drawHud(void);
int initGame(void);
void newGame(void);
void gameShutdown(void);
void gameLoop(void);
void drawPlay(void);
void pickAvailableCargo(void);
void getPlayerInput(void);
void drawGameMenu(void);
void showManifest(void);
void buyCargo(void);
void popupInfo(char *, int);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
void starportMap(void);
void travelSelect(void);
void endGame(void);
static inline int rollRand(int, int);
bool portCargoExists(int);
int getAvailableBay(void);
int totalTonnage(void);
void appraiseCargo(void);
void crewStrike(void);
void strikeCheck(void);
void renewLease(void);
void updatePlayer(void);
void inspectShip(void);
void repairShip(void);
char *formatLong(long);
void upgradeShip(void);
void retire(void);
bool cargoInPort(int);
void getPortCargoInfo(int, int *);
int availablePortSlot(void);
void calculateMap(void);

/*
    Recalculate distance and direction
    of all starports
*/
void calculateMap() {
    int i;
    float X2, Y2;

    for (i = 0; i < MAX_STARPORTS; i++) {
        if (i != cur_starport) {
            X2 = starports[i].X1 - starports[cur_starport].X1;
            Y2 = starports[i].Y1 - starports[cur_starport].Y1;
            /* If we want to show the distance in parsecs instead of light-years
               we divide the distance calculated by 3.26156.
             */
            starports[i].distance = sqrt(SQR(X2) + SQR(Y2));

            /* Direction is a little trickier */
            if (X2 != 0) {
                starports[i].direction = atan(Y2/X2) * 180 / M_PI;
            } else {
                if (Y2 > 0) {
                    starports[i].direction = 90;
                } else {
                    starports[i].direction = 270;
                }
            }

            if (X2 < 0) {
                starports[i].direction += 180;
            }

            if (starports[i].direction > 360) {
                starports[i].direction -= 360;
            } else if (starports[i].direction < 0) {
                starports[i].direction += 180;
            }
        }
    }
}
```

/* Listing continued on next page…*/

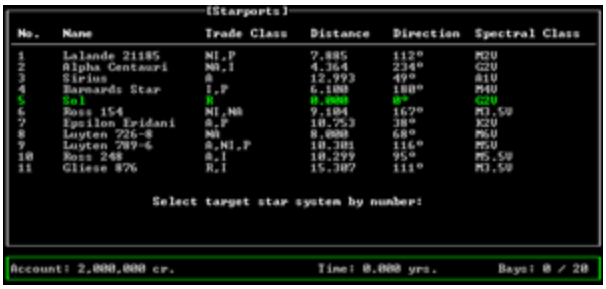
/* Listing continued from previous page */

```
int availablePortSlot() {
    int i;

    for (i = 0; i < MAX_CARGO_FOR_SALE; i++) {
        if (starport_cargo[i].cargo_type == -1) return i;
    }
    return -1;
}

void getPortCargoInfo(int cargoIdx, int *pct) {
    int i;

    for (i = 0; i < MAX_CARGO_FOR_SALE; i++) {
        if (starport_cargo[i].cargo_type == cargoIdx) {
            *pct = starport_cargo[i].pct;
            return;
        }
    }
}

/*
    Determines if a particular cargo type is
    in port.  This is used to synchronize the market
    rates of a piece of cargo onboard against what
    is at a particular starport.
*/
bool cargoInPort(int cargo_type) {
    int i;

    for (i = 0; i < MAX_CARGO_FOR_SALE; i++) {
        if (starport_cargo[i].cargo_type == cargo_type) return TRUE;
    }

    return FALSE;
}

void retire() {
    wclear(win);

    wresize(win,height,width);
    wbkgd(win,COLOR_PAIR(1));
    box(win,ACS_VLINE,ACS_HLINE);
    wattrset(win,A_BOLD);
    mvwprintw(win,0,27,"YOU WIN!!!");
    wattrset(win,A_NORMAL);
    mvwprintw(win,2,5,"You have successfully made it to retirement as a Star Mer-
chant!");
    mvwprintw(win,6,20,"Earnings: %s cr.",formatLong(bankAccount));
    mvwprintw(win,7,20,"Game Time: %3.3f years",shipTime);
    mvwprintw(win,8,20,"Light-years traveled: %3.3f ly.", totalDistance);

    wattrset(win,A_REVERSE);
    mvwprintw(win,10,28,"Press any key");
    wattrset(win,A_NORMAL);

    wgetch(win);
    game_state = GAME_DONE;
    quit = TRUE;
    /*gameShutdown();*/
}

void upgradeShip() {
    int key;
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    int i;
    int startY = 6;

    wclear(win);
    box(win,ACS_VLINE,ACS_HLINE);
    mvwprintw(win,0,27,"[%s Shipyards]",starports[cur_starport].port_name);

    wattrset(win,A_BOLD);
    mvwprintw(win,2,20,"Select your upgrade below:");
    wattrset(win,A_NORMAL);

    wrefresh(win);
    wattrset(win,A_BOLD);
    mvwprintw(win,4,4,"No.\tDescription");
    mvwprintw(win,4,35,"Speed");
    mvwprintw(win,4,42,"Tonnage");
    mvwprintw(win,4,50,"Price");

    mvwhline(win,5,1,ACS_HLINE,width-2);
    wattrset(win,A_NORMAL);

    for (i = 0; i < MAX_UPGRADES; i++) {
        if (i == ship_level) {
            wattrset(win,COLOR_PAIR(11) | A_BOLD);
        } else {
            wattrset(win,A_NORMAL);
        }
        mvwprintw(win,startY,4,"%i.",i+1);
        mvwprintw(win,startY,8,"%s",upgrades[i].package_name);

        mvwprintw(win,startY,35,"%1.0fc",(C_MS/(upgrades[i].engine_speed*C_MS)));
        mvwprintw(win,startY,42,"%i",upgrades[i].tonnage);
        mvwprintw(win,startY,50,"%s cr.",formatLong(upgrades[i].upgrade_cost));
        startY++;
    }

    wattrset(win,A_NORMAL);
    echo();
    curs_set(2);
    mvwprintw(win,17,17,"Which upgrade would you like: ");
    do {
        key = wgetch(win);
    } while ( (key - 48 < 0) && (key - 48 > MAX_UPGRADES) && (key != KEY_ENTER));
    noecho();
    curs_set(0);
    if (key - 48 < 0 || key - 48 > MAX_UPGRADES) return;
    key -= 48;
    if (key == 0 || key == 10 || key == 13 || key == KEY_ENTER) return;
    key--;

    if (key == ship_level) {
        popupInfo("Ship already has that package",RED_PAIR);
        return;
    }

    if (key < ship_level) {
        popupInfo("You cannot downgrade packages",RED_PAIR);
        return;
    }

    if (upgrades[key].upgrade_cost >= bankAccount) {
        popupInfo("Insufficient credits",RED_PAIR);
        return;
```



/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    }

    ship_level = key;
    ship_tonnage = upgrades[key].tonnage;
    bankAccount -= upgrades[key].upgrade_cost;
    elapsedTime += 0.04;
    shipTime += 0.04;
    drawHud();
    popupInfo("Your ship has been upgraded",CYAN_PAIR);
}

void repairShip() {
    int key;
    long repair_cost = 0;
    float repair_time = 0.00;
    wclear(win);

    repair_time = 0.03 * ship_damage;
    repair_cost = (bankAccount * 0.025) * ship_damage;
    wattrset(win,COLOR_PAIR(RED_PAIR));
    box(win,ACS_BOARD,ACS_BOARD);
    wattrset(win,A_BOLD);
    mvwprintw(win,0,27," %s Drydock ",starports[cur_starport].port_name);
    wattrset(win,A_NORMAL);

    mvwprintw(win,2,10,"Scans show your ship has sustained %i%% damage to its
hull",ship_damage);
    mvwprintw(win,4,10,"Repair cost: %s cr.",formatLong(repair_cost));
    mvwprintw(win,5,10,"Repair time: %2.3f years",repair_time);

    noecho();
    wattrset(win,A_BOLD);
    mvwprintw(win,8,30,"Repair Ship? (Y/N)");
    wattrset(win,A_NORMAL);

    do {
        key = wgetch(win);
    } while ((tolower(key) != 'q')
      &&(tolower(key) != 'n')&&(tolower(key) != 'y'));

    key = tolower(key);
    if (key == 'y') {
        bankAccount -= repair_cost;
        shipTime += repair_time;
        elapsedTime += repair_time;
        ship_damage = 0;
        drawHud();
        popupInfo("Your ship has been repaired",CYAN_PAIR);
        strikeCheck();
        updatePlayer();
    }
}

char *formatLong(long num) {
    static int comma = '\0';
    static char retbuf[30];
    char *p = &retbuf[sizeof(retbuf)-1];
    int i = 0;

    if(comma == '\0') {
        struct lconv *lcp = localeconv();
        if(lcp != NULL) {
            if(lcp->thousands_sep != NULL &&
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
                *lcp->thousands_sep != '\0')
            comma = *lcp->thousands_sep;
            else comma = ',';
        }
    }

    *p = '\0';

    do {
        if(i%3 == 0 && i != 0)
        *--p = comma;
        *--p = '0' + num % 10;
        num /= 10;
        i++;
    } while(num != 0);

    return p;
}

/*
    Ship status report
*/
void inspectShip() {
    wclear(shipstatus);
    wbkgd(shipstatus,COLOR_PAIR(3));
    box(shipstatus,ACS_VLINE,ACS_HLINE);
    wattrset(shipstatus,A_BOLD);
    mvwprintw(shipstatus,0,27,"[Ship Status]");

    mvwprintw(shipstatus,2,10,"Cargo Capacity: %i tons", ship_tonnage);
    mvwprintw(shipstatus,3,11,"Capacity Used: %i tons", totalTonnage());
    mvwprintw(shipstatus,4,14,"Cargo Bays: %i in use, %i available",
      MAX_CARGO_PARTITIONS - partitions_available, partitions_available);
    mvwprintw(shipstatus,5,13,"Hull Damage: %i%%",ship_damage);
    mvwprintw(shipstatus,6,10,"Total Distance: %4.3f ly",totalDistance);
    mvwprintw(shipstatus,7,10,"Engine Package: %s",
      upgrades[ship_level].package_name);

    mvwprintw(shipstatus,10,13,"Crew Salary: %s cr./yr.",
      formatLong(crew_pay_rate));
    mvwprintw(shipstatus,11,4,"Lease time remaining: %1.3f years",
      2.00 - elapsedTime);

    if (ship_damage > 0) {
        wattrset(shipstatus,A_BOLD);
        mvwprintw(shipstatus,13,5,
      "Current damage will result in %2.3f years extra per light-year of travel",
      ship_damage * DAMAGE_SLOWDOWN);
        wattrset(shipstatus,A_NORMAL);
    }

    wattrset(shipstatus,A_REVERSE);
    mvwprintw(shipstatus,20,25,"Press the any key");
    wattrset(shipstatus,A_NORMAL);
    wgetch(shipstatus);
}

void renewLease() {
    int key;
    wclear(win);
    wbkgd(win,COLOR_PAIR(1));
    box(win,ACS_VLINE,ACS_HLINE);
    wattrset(win,A_BOLD);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    mvwprintw(win,1,25,"Your lease has expired");
    wattrset(win,A_NORMAL);

    if (bankAccount < LEASE_COST) {
        wattrset(win,A_BOLD);
        mvwprintw(win,8,20,"You do not have enough in your account to renew.");
        mvwprintw(win,10,20,"Play Again? (Y/N)");
        do {
          key = wgetch(win);
        } while ((tolower(key) != 'q')
          &&(tolower(key) != 'n')&&(tolower(key) != 'y'));

        if (key == 'n' || key == 'q') {
          quit = TRUE;
        } else {
          game_state = GAME_INTRO;
          num_runs++;
          newGame();
        }
    } else {
        mvwprintw(win,10,20,"Renew your lease? (Y/N)");
        do {
            key = wgetch(win);
        } while ((tolower(key) != 'q')&&(tolower(key) != 'n')&&(tolower(key) !=
'y'));

        if (key == 'n' || key == 'q') {
            quit = TRUE;
        } else {
            bankAccount -= LEASE_COST;
            elapsedTime = 0.00;
            drawHud();
        }
    }
}

void crewStrike() {
    short num_attempts = 0;
    char input[11];
    long asking_price;
    long counter_offer;
    long crew_floor;
    bool offer_accepted = FALSE;

    asking_price = crew_pay_rate + (0.5 * rollRand(1,game_seed)) * crew_pay_rate;

    popupInfo("The crew has gone on strike!",RED_PAIR);
    echo();
    while (offer_accepted == FALSE) {
        wclear(win);
        box(win,ACS_VLINE,ACS_HLINE);
        wattrset(win,A_BOLD | COLOR_PAIR(11));
        mvwprintw(win,0,27,"[Salary Negotiatron 5000]");
        mvwprintw(win,2,20,"The crew has gone on strike!");
        wattrset(win,A_NORMAL);

        mvwprintw(win,2,10,
          "You are currently paying the crew %s credits annually",
          formatLong(crew_pay_rate));
        mvwprintw(win,4,10,"The crew is asking for a pay rate of %s credits",
          formatLong(asking_price));
        curs_set(2);
        mvwprintw(win,6,10,"Enter your counter-offer: ");
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
        memset(input,0,sizeof(input));
        wgetnstr(win,input,10);
        curs_set(0);
        counter_offer = atoi(input);
        crew_floor = crew_pay_rate+num_attempts*(asking_price-crew_pay_rate/10);
        if ((counter_offer >= asking_price) || (counter_offer >= crew_floor)) {
            crew_pay_rate = counter_offer;
            offer_accepted = TRUE;
            popupInfo("Offer Accepted!",CYAN_PAIR);
        } else {
            popupInfo("Offer Rejected!",RED_PAIR);
            offer_accepted = FALSE;
            num_attempts++;
        }
    }
    noecho();
}

void strikeCheck() {
    int roll;

    roll = rollRand(1,game_seed);
    if (roll < (.035 * (bankAccount / crew_pay_rate))) crewStrike();
}

/*
    Appraises all cargo coming into
    a Starport to assess its local value
*/
void appraiseCargo() {
    int i,j;
    int D2,D3;
    float pct;
    int tmp;

    D2 = starports[cur_starport].trade;
    pct = 0.00;
    for (i = 0; i < 6; i++) {
        D3 = pow(2,(6 - i + 1));
        trade_factors[i] = (int)(D2 / D3);
    }

    for (i = 0; i < MAX_CARGO_PARTITIONS; i++) {
        if (cargo_bays[i].cargo_type != -1) {
            if (cargoInPort(cargo_bays[i].cargo_type)) {
                getPortCargoInfo(cargo_bays[i].cargo_type,&tmp);
                cargo_bays[i].pct = tmp;
                cargo_bays[i].price = cargos[cargo_bays[i].cargo_type].basePrice
                  * cargo_bays[i].amt * (cargo_bays[i].pct * 0.01);
            } else {
                D2 = 0;
                for (j = 0; j < 6; j++) {
                    D2 += cargos[cargo_bays[i].cargo_type].price_factors[j]
                      * trade_factors[j];
                }
                D3 = rollRand(0,6) + rollRand(0,6) + D2 + 1;
                if (D3 < 0) D3 = 0;
                if (D3 >= PRICE_DATA_MAX) D3 = PRICE_DATA_MAX - 1;

                pct = price_data[D3];
               #ifdef PDCURSES
                       cargo_bays[i].pct = (int)round(pct * 100);
               #else
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
                        cargo_bays[i].pct = (int)floor(pct * 100);
                #endif
                 cargo_bays[i].price = cargos[cargo_bays[i].cargo_type].basePrice
                   * cargo_bays[i].amt * (cargo_bays[i].pct * 0.01);
            }
            cargo_bays[i].in_port = FALSE;
        }
    }
}

int totalTonnage() {
    int i;
    int total = 0;

    for (i = 0; i < MAX_CARGO_PARTITIONS; i++) {
        if (cargo_bays[i].cargo_type != -1)
            total += cargo_bays[i].amt;
    }

    return total;
}

int getAvailableBay() {
    int i;

    for (i = 0; i < MAX_CARGO_PARTITIONS; i++) {
        if (cargo_bays[i].cargo_type == -1) return i;
    }

    return -1;
}

void endGame() {
    int key;

    wclear(win);
    wbkgd(win,COLOR_PAIR(1));
    box(win,ACS_VLINE,ACS_HLINE);

    wattrset(win,A_BOLD);
    mvwprintw(win,3,25,"Game Over");
    mvwprintw(win,9,18,"Account Balance: %s credits",formatLong(bankAccount));
    mvwprintw(win,12,20,"Play Again? (Y/N)");
    wattrset(win,A_NORMAL);

    do {
        key = wgetch(win);
    } while ((tolower(key)!='q')&&(tolower(key)!='n')&&(tolower(key)!='y'));

    if (key == 'n' || key == 'q') {
        quit = TRUE;
    } else {
        quit = FALSE;
        game_state = GAME_INTRO;
        num_runs++;
        newGame();
    }
}

void travelSelect() {
    char input[3];
    int key;
    long fuel_expense;
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    float travelTime;
    long crew_travel_pay;

    starportMap();
    flushinp();
    echo();
    wattrset(win,A_BOLD);
    mvwprintw(win,17,20,"Select target star system by number: ");
    curs_set(2);
    wattrset(win,A_NORMAL);
    wgetnstr(win,input,2);
    curs_set(0);
    wrefresh(win);

    key = atoi(input);

    if (key - 1 == cur_starport) {
        popupInfo("You are at that starport",RED_PAIR);
        return;
    }

    if (key == 0) return;

    if (key > MAX_STARPORTS) {
        popupInfo("Invalid selection",RED_PAIR);
        return;
    }

    noecho();
    key--;
    cur_starport = key;
    flash();

    /*
        Perform the necessary calculations here
        for docking, fuel costs, crew salary, etc.
        Travel time now includes damage, which can
        slow a ship down
    */
    travelTime = ( ((DAMAGE_SLOWDOWN * ship_damage) * starports[key].distance)
      * starports[key].distance) + (upgrades[ship_level].engine_speed
      * starports[key].distance) + upgrades[ship_level].engine_speed;
    totalDistance += starports[key].distance;

    /* Check for damage to ship in transit */
    if (rollRand(1, game_seed) < (.25*(totalDistance/starports[key].distance))) {
        if (ship_damage < 100) ship_damage += rollRand(1,10);
        if (ship_damage > 100) ship_damage = 100;
    }

    fuel_expense = starports[key].distance * FUEL_COST;
    elapsedTime += travelTime;
    shipTime += travelTime;
    crew_travel_pay = travelTime * crew_pay_rate;

    bankAccount -= DOCKING_FEE + fuel_expense + crew_travel_pay;
    if (bankAccount < 0) bankAccount = 0;
    drawHud();

    wclear(win);
    box(win,ACS_VLINE,ACS_HLINE);

    wattrset(win,A_REVERSE);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    mvwprintw(win,0,26,"Arrival Report - %s",starports[key].port_name);
    wattrset(win,A_NORMAL);

    mvwprintw(win,3,4,"Expenses have been deducted as follows:");
    mvwprintw(win,5,10,"Docking fee:");
    mvwprintw(win,5,30,"%s cr.",formatLong(DOCKING_FEE));
    mvwprintw(win,6,10,"Fuel:");
    mvwprintw(win,6,30,"%s cr.", formatLong(fuel_expense));
    mvwprintw(win,7,10,"Crew salary: ");
    mvwprintw(win,7,30,"%s cr.", formatLong(crew_travel_pay));
    mvwhline(win,8,10,ACS_HLINE,35);
    mvwprintw(win,9,10,"Total:");
    mvwprintw(win,9,30,"%s cr.",
      formatLong(crew_travel_pay+fuel_expense+DOCKING_FEE));

    mvwprintw(win,11,10,"Total Travel Time:");
    mvwprintw(win,11,30,"%2.3f years",travelTime);

    mvwprintw(win,13,10,"Your cargo has been appraised by our autoscanner");

    if (ship_damage > 0) {
        wattrset(win,COLOR_PAIR(RED_PAIR) | A_BLINK);
        mvwprintw(win,15,18,"Docking sensors report damage to the hull");
        wattrset(win,A_NORMAL);
    }

    if (bankAccount <= 0) {
        endGame();
    } else {
        pickAvailableCargo();
        appraiseCargo();
        wattrset(win, A_REVERSE);
        mvwprintw(win,20,20,"Press any key to enter %s",
          starports[key].port_name);
        wattrset(win,A_NORMAL);
        wrefresh(win);
        wgetch(win);
        strikeCheck();
        if (elapsedTime > 2) renewLease();
        calculateMap();
    }
}

void starportMap() {
    int i;
    int startY = 4;

    wclear(win);
    wbkgd(win,COLOR_PAIR(0));
    box(win,ACS_VLINE,ACS_HLINE);

    wattrset(win,A_BOLD);
    mvwprintw(win,0,27,"[Starports]");

    mvwprintw(win,2,2,"No.\tName");
    mvwprintw(win,2,27,"Trade Class");
    mvwprintw(win,2,41,"Distance");
    mvwprintw(win,2,52,"Direction");
    mvwprintw(win,2,63,"Spectral Class");
    mvwhline(win,3,1,ACS_HLINE,width-2);

    for (i = 0; i < MAX_STARPORTS; i++) {
        if (i == cur_starport) {
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
            wattrset(win,COLOR_PAIR(11) | A_BOLD);
            mvwprintw(win,startY,2,"%i",i+1);
            mvwprintw(win,startY,8,"%s",starports[i].port_name);
            mvwprintw(win,startY,27,"%s",starports[i].classification);
            mvwprintw(win,startY,41,"%1.3f",0.00);
            mvwprintw(win,startY,52,"%i",0);
            waddch(win,ACS_DEGREE);
            mvwprintw(win,startY,63,"%s",starports[i].spectral_type);
            wattrset(win,A_NORMAL);
        } else {
            wattrset(win,A_NORMAL);
            mvwprintw(win,startY,2,"%i\t%s",i+1,starports[i].port_name);
            mvwprintw(win,startY,27,"%s",starports[i].classification);
            mvwprintw(win,startY,41,"%1.3f",starports[i].distance);
            mvwprintw(win,startY,52,"%i",starports[i].direction);
            waddch(win,ACS_DEGREE);
            mvwprintw(win,startY,63,"%s",starports[i].spectral_type);
        }
        startY++;
    }
    wattrset(win,A_NORMAL);
    wrefresh(win);
}

void popupInfo(char *msg, int display_color) {
    wclear(infowin);
    wresize(infowin,3,strlen(msg)+2);
    wbkgd(infowin,COLOR_PAIR(display_color) | A_BOLD);
    wattrset(infowin,A_NORMAL);
    box(infowin,ACS_VLINE,ACS_HLINE);
    wattrset(infowin,A_BOLD);
    mvwprintw(infowin,1,1,msg);
    wattrset(infowin,A_NORMAL);
    wrefresh(infowin);
    napms(1300);
}

void sellCargo() {
    char input[3];
    int key;
    int i;
    int slot;

    showManifest();
    if (partitions_available == MAX_CARGO_PARTITIONS) {
        popupInfo("You have no cargo to sell",RED_PAIR);
        return;
    }
    flushinp();
    echo();
    curs_set(2);
    mvwprintw(manifest,20,17,"Select cargo bay to sell: ");
    memset(input,0,sizeof(input));
    wgetnstr(manifest,input,2);
    curs_set(0);
    box(manifest,ACS_VLINE,ACS_HLINE);
    wattrset(manifest,A_BOLD);
    mvwprintw(manifest,0,27,"[Ship Manifest]");
    wattrset(manifest,A_NORMAL);
    wrefresh(manifest);

    if (tolower(input[0]) == 'a') {
        for (i = 0; i < MAX_CARGO_PARTITIONS; i++) {
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
            if (cargo_bays[i].cargo_type != -1) {
                bankAccount += cargo_bays[i].price;
                partitions_available++;
                shipTime += .003;
                elapsedTime += .003;

                /* If you're buying this at the same port and haven't
                    left yet, it goes back on the shelf */
                if (cargo_bays[i].in_port == TRUE) {
                    slot = availablePortSlot();
                    starport_cargo[slot].cargo_type = cargo_bays[i].cargo_type;
                    starport_cargo[slot].pct = cargo_bays[i].pct;
                    starport_cargo[slot].amt = cargo_bays[i].amt;
                    starport_cargo[slot].price = cargo_bays[i].price;
                    cargo_bays[i].in_port = FALSE;
                }
                cargo_bays[i].cargo_type = -1;
            }
        }

        drawHud();
        popupInfo("All cargo sold.  Thank you!",CYAN_PAIR);
    } else {
        key = atoi(input);
        if (key == 0) return;
        key--;

        if (cargo_bays[key].cargo_type == -1) {
            popupInfo("Cargo bay is empty",RED_PAIR);
            return;
        }

        /* If you're buying this at the same port and haven't
            left yet, it goes back on the shelf */
        if (cargo_bays[key].in_port == TRUE) {
            slot = availablePortSlot();
            starport_cargo[slot].cargo_type = cargo_bays[key].cargo_type;
            starport_cargo[slot].pct = cargo_bays[key].pct;
            starport_cargo[slot].amt = cargo_bays[key].amt;
            starport_cargo[slot].price = cargo_bays[key].price;
            cargo_bays[key].in_port = FALSE;
        }

        bankAccount += cargo_bays[key].price;
        partitions_available++;
        shipTime += .003;
        elapsedTime += .003;
        cargo_bays[key].cargo_type = -1;
        drawHud();
        popupInfo("Thank you for your business",CYAN_PAIR);
    }

    strikeCheck();
    updatePlayer();
}

void buyCargo() {
    int key;
    int i;
    int startY = 4;
    int tmp = -1;

    flushinp();
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    wclear(win);
    box(win,ACS_VLINE,ACS_HLINE);
    wattrset(win,A_BOLD);
    mvwprintw(win,0,27,"[Buy Cargo at %s]",starports[cur_starport].port_name);
    wattrset(win,A_NORMAL);

    if (partitions_available == 0) {
        popupInfo("All cargo bays are full",RED_PAIR);
        return;
    }

    if (port_cargos_remaining == 0) {
        popupInfo("All cargo in port has been sold",RED_PAIR);
        return;
    }

    wrefresh(win);
    wattrset(win,A_BOLD);
    mvwprintw(win,2,4,"No.\tDescription");
    mvwprintw(win,2,31,"Tons");
    mvwprintw(win,2,38,"Price");
    mvwprintw(win,2,53,"%%");

    mvwhline(win,3,1,ACS_HLINE,width-2);
    wattrset(win,A_NORMAL);

    for (i = 0; i < MAX_CARGO_FOR_SALE; i++) {
        if (starport_cargo[i].cargo_type != -1) {
            mvwprintw(win,startY,4,"%i.",i+1);
            mvwprintw(win,startY,8,"%s",cargos[starport_cargo
[i].cargo_type].cargo_name);
            mvwprintw(win,startY,31,"%i",starport_cargo[i].amt);
            mvwprintw(win,startY,38,"%s",formatLong(starport_cargo[i].price));
            mvwprintw(win,startY,53,"%i",starport_cargo[i].pct);
        } else {
            mvwprintw(win,startY,4,"%i.",i+1);
            wattrset(win,COLOR_PAIR(4) | A_DIM);
            mvwprintw(win,startY,8,"EMPTY");
            wattrset(win,A_NORMAL);
        }
        startY++;
    }
    echo();

    mvwprintw(win,17,17,"Select lot to purchase by number: ");
    curs_set(2);
    do {
        key = wgetch(win);
    } while ((key-48 < 0)&&(key-48 > MAX_CARGO_FOR_SALE) && (key != KEY_ENTER));
    noecho();
    curs_set(0);
    if (key - 48 < 0 || key - 48 > MAX_CARGO_FOR_SALE) return;
    key -= 48;

    if (key == 0 || key == 10 || key == 13 || key == KEY_ENTER) return;
    key--;

    if (starport_cargo[key].cargo_type == -1) {
        popupInfo("Lot is empty",RED_PAIR);
        return;
    }

    if (starport_cargo[key].price >= bankAccount) {
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
        popupInfo("You don't have enough credits",RED_PAIR);
        return;
    }

    if (totalTonnage() + starport_cargo[key].amt > ship_tonnage) {
        popupInfo("Tonnage exceeds ship limit",RED_PAIR);
        return;
    }

    if (partitions_available == 0) {
        popupInfo("You have no bays available",RED_PAIR);
        return;
    }

    tmp = getAvailableBay();
    cargo_bays[tmp].cargo_type = starport_cargo[key].cargo_type;
    cargo_bays[tmp].amt = starport_cargo[key].amt;
    cargo_bays[tmp].price = starport_cargo[key].price;
    cargo_bays[tmp].pct = starport_cargo[key].pct;
    cargo_bays[tmp].purchase_loc = cur_starport;
    cargo_bays[tmp].in_port = TRUE;
    partitions_available--;

    bankAccount -= starport_cargo[key].price;
    starport_cargo[key].cargo_type = -1;
    shipTime += 0.003;
    elapsedTime += 0.003;

    drawHud();
    popupInfo("Thank you for your purchase!",CYAN_PAIR);
    strikeCheck();
    updatePlayer();
}


/*  Shows the ship's manifest of cargo */
void showManifest() {
    int i;
    int startY = 4;

    wclear(manifest);
    wbkgd(manifest,COLOR_PAIR(0));
    box(manifest,ACS_VLINE,ACS_HLINE);
    wattrset(manifest,A_BOLD);
    mvwprintw(manifest,0,27,"[Ship Manifest]");

    if (partitions_available == MAX_CARGO_PARTITIONS) {
        mvwprintw(manifest,3,20,"You have no cargo onboard");
    } else {
        mvwprintw(manifest,2,4,"Bay\tDescription");
        mvwprintw(manifest,2,31,"Tons");
        mvwprintw(manifest,2,38,"Price");
        mvwprintw(manifest,2,50,"%%");
        mvwprintw(manifest,2,56,"Purchased At");
        mvwhline(manifest,3,1,ACS_HLINE,width-2);
        wattrset(manifest,A_NORMAL);

        for (i = 0; i < MAX_CARGO_PARTITIONS; i++) {
            if (cargo_bays[i].cargo_type != -1) {
                mvwprintw(manifest,startY,4,"%i.\t%s",i+1,
                  cargos[cargo_bays[i].cargo_type].cargo_name);
                mvwprintw(manifest,startY,31,"%i",cargo_bays[i].amt);
                mvwprintw(manifest,startY,38,"%s",
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
                  formatLong(cargo_bays[i].price));
                mvwprintw(manifest,startY,50,"%i",cargo_bays[i].pct);
                mvwprintw(manifest,startY,56,"%s",
                  starports[cargo_bays[i].purchase_loc].port_name);
                startY++;
            }
        }
    }
}

void drawGameMenu() {
    noecho();
    wclear(win);
    box(win,ACS_VLINE,ACS_HLINE);

    wattrset(win,A_BOLD | COLOR_PAIR(11));
    mvwprintw(win,0,27,"[Welcome to %s]",starports[cur_starport].port_name);

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,5,27,"B");
    wattrset(win,A_NORMAL);
    mvwprintw(win,5,28,"uy Cargo");

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,6,27,"S");
    wattrset(win,A_NORMAL);
    mvwprintw(win,6,28,"ell Cargo");

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,7,27,"L");
    wattrset(win,A_NORMAL);
    mvwprintw(win,7,28,"ist Ship Manifest");

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,8,27,"M");
    wattrset(win,A_NORMAL);
    mvwprintw(win,8,28,"ap of Starports");

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,9,27,"T");
    wattrset(win,A_NORMAL);
    mvwprintw(win,9,28,"ravel to Starport");

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,10,27,"I");
    wattrset(win,A_NORMAL);
    mvwprintw(win,10,28,"nspect Ship");

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,11,27,"U");
    wattrset(win,A_NORMAL);
    mvwprintw(win,11,28,"pgrade Ship");

    if (ship_damage > 0) {
        wattrset(win,COLOR_PAIR(10) | A_BOLD);
        mvwprintw(win,12,27,"R");
        wattrset(win,A_NORMAL);
        mvwprintw(win,12,28,"epair Ship");
    }

    wattrset(win,COLOR_PAIR(10) | A_BOLD);
    mvwprintw(win,14,27,"Q");
    wattrset(win,A_NORMAL);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    mvwprintw(win,14,28,"uit");

    wattrset(win,A_NORMAL);
    wrefresh(win);
}

void getPlayerInput() {
    int key;

    flushinp();
    keypad(win, TRUE);
    raw();

    key = wgetch(win);

    switch(key) {
        case 'u':
        case 'U':
            upgradeShip();
            break;
        case 'r':
        case 'R':
            if (ship_damage == 0) {
                popupInfo("Your ship doesn't need any repairs",RED_PAIR);
            } else {
                repairShip();
            }
            break;
        case 'i':
        case 'I':
            inspectShip();
            break;
        case 'b':
        case 'B':
            buyCargo();
            break;
        case 's':
        case 'S':
            sellCargo();
            break;
        case 't':
        case 'T':
            if (ship_damage == 100) {
                popupInfo("Your ship is too damaged",RED_PAIR);
            } else {
                travelSelect();
            }
            break;
        case 'l':
        case 'L':
            showManifest();
            wattrset(manifest,A_REVERSE);
            mvwprintw(manifest,height-5,20,"Press any key");
            wattrset(manifest,A_NORMAL);
            wrefresh(manifest);
            wgetch(manifest);
            break;
        case 'm':
        case 'M':
            starportMap();
            wattrset(win,A_REVERSE);
            mvwprintw(win,20,27,"Press any key");
            wattrset(win,A_NORMAL);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
            wgetch(win);
            break;
        case 'Q':
        case 'q':
            quit = TRUE;
            game_state = GAME_DONE;
            break;
        default:
            wattrset(win,A_REVERSE | COLOR_PAIR(RED_PAIR));
            mvwprintw(win,3,27,"Invalid command");
            wattrset(win,A_NORMAL);
            wrefresh(win);
            napms(750);
            break;
    }
}

static inline int rollRand(int min, int max) {
    double r;
    double x;
    r = ((double)rand() / ((double)(RAND_MAX)+(double)(1)));
        x = (r * max);
    return (int)x + min;
}

/*  Prevents duplicates from appearing
    in the Buy Cargo screen
*/
bool portCargoExists(int cargoIdx) {
    int i;

    for (i = 0; i < MAX_CARGO_FOR_SALE; i++) {
        if (starport_cargo[i].cargo_type == cargoIdx) return TRUE;
    }
    return FALSE;
}

/* Randomly selects a cargo available at a starport */
void pickAvailableCargo() {
    int i = 0;
    int j;
    int Q2,Q3;
    int D3;
    int D2;
    float pct;
    int cargoIdx;

    port_cargos_remaining = MAX_CARGO_FOR_SALE;

    D2 = starports[cur_starport].trade;

    for (i = 0; i < 6; i++) {
        D3 = pow(2,(6 - i + 1));
       #ifdef PDCURSES
               trade_factors[i] = (int)round((D2 / D3));
       #else
               trade_factors[i] = (int)floor((D2 / D3));
       #endif
    }

    i = 0;
    /* Reset all of the cargo at the port to empty first */
    while (i < MAX_CARGO_FOR_SALE) {
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
        starport_cargo[i].cargo_type = -1;
        cargoIdx = rollRand(0,MAX_CARGOS);
        if (cargoIdx == MAX_CARGOS) cargoIdx = MAX_CARGOS - 1;
        if (!portCargoExists(cargoIdx)) {
            starport_cargo[i].cargo_type = cargoIdx;
            D2 = 0;
            for (j = 0; j < 6; j++) {
                D2 += cargos[i].price_factors[j] * trade_factors[j];
            }
            D3 = rollRand(0,6) + rollRand(0,6) + D2 + 1;
            if (D3 < 0) D3 = 0;
            if (D3 >= PRICE_DATA_MAX) D3 = PRICE_DATA_MAX - 1;

            pct = price_data[D3];
           #ifdef PDCURSES
              starport_cargo[i].pct = (int)round(pct * 100);
           #else
              starport_cargo[i].pct = (int)floor(pct * 100);
           #endif

            Q2 = 1;
            if (cargoIdx > 18) Q2 = 5;
            if (cargoIdx > 33) Q2 = 10;

            Q3 = 0;
            for (j = 0; j < cargos[cargoIdx].amt_multiplier; j++) {
                Q3 += rollRand(1,6) * Q2;
            }

            if (Q3 > ship_tonnage) Q3 = ship_tonnage;
            starport_cargo[i].amt = Q3;
            starport_cargo[i].price = cargos[cargoIdx].basePrice * Q3 *
(starport_cargo[i].pct * 0.01);
            i++;
        }
    }
}

void drawPlay() {
    wclear(win);
    wbkgd(win,COLOR_PAIR(0));
    box(win,ACS_VLINE,ACS_HLINE);
    wrefresh(win);
}

void displayMessage(WINDOW *w, int y, int x, int lines, char **msg) {
  int c;

  for (c = 0; c < lines; c++)
    mvwprintw (w, y + c, x, msg[c]);
}

/* Draws the HUD (Heads-Up Display) */
void drawHud() {
    wclear(hud);
    wattrset(hud,COLOR_PAIR(11) | A_BOLD);
    box(hud,ACS_VLINE,ACS_HLINE);
    wattrset(hud,A_NORMAL);
    mvwprintw(hud,1, 1, "Account: %s cr.", formatLong(bankAccount));
    mvwprintw(hud,1, 42, "Time: %3.3f yrs.", shipTime);
    mvwprintw(hud,1, 66, "Bays: %i / %i",
      MAX_CARGO_PARTITIONS - partitions_available, MAX_CARGO_PARTITIONS);
    wrefresh(hud);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
}

/*
    Starts a new game
    by initializing specific instance
    data and then setting the state to GAME_PLAYING
*/
void newGame() {
    int i;

    bankAccount = 2000000;                    /* Start with 2 million credits */
    shipTime = 0.00;                    /* Internal ship time, starts at 0.00 */
    elapsedTime = 0.00;
    cur_starport = 4;                                    /* Start off at Sol */
    partitions_available = MAX_CARGO_PARTITIONS;/* Start with an unladen ship */
    game_seed = rollRand(1,500);
    crew_pay_rate = CREW_STARTING_PAY;
    ship_tonnage = MAX_TONS;
    ship_damage = 0;
    totalDistance = 0.00;
    ship_level = 0;
    pickAvailableCargo();
    calculateMap();

    for (i = 0; i < MAX_CARGO_PARTITIONS; i++) {
        cargo_bays[i].cargo_type = -1;
    }

    wclear(win);
    wresize(win,height,width);
    box(win,ACS_VLINE,ACS_HLINE);
    wbkgd(win,COLOR_PAIR(1) | A_DIM);

    if (num_runs == 0) {
        /*
            Show the intro text
        */
        wattrset(win,A_BOLD);
        mvwprintw(win,2,25,"Welcome to Star Merchant");
        wattrset(win,A_NORMAL);

        displayMessage(win,4,3,2,intro_msg);
        displayMessage(win,7,3,3,intro_msg2);
        displayMessage(win,11,3,2,intro_msg3);
        displayMessage(win,14,3,2,intro_msg4);

        wattrset(win,A_REVERSE);
        mvwprintw(win,20,25,"Press any key to continue...");
        wattrset(win,A_NORMAL);
        wgetch(win);

        wclear(win);
        box(win,ACS_VLINE,ACS_HLINE);
        wbkgd(win,COLOR_PAIR(1) | A_DIM);

        wattrset(win,A_BOLD);
        mvwprintw(win,2,25,"Starport Trade Classification");
        wattrset(win,A_NORMAL);

        displayMessage(win,4,3,2,intro_msg5);
        displayMessage(win,7,3,1,intro_msg6);
        displayMessage(win,9,10,6,intro_msg7);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
        wattrset(win,A_REVERSE);
        mvwprintw(win,20,25,"Press any key to start");
        wattrset(win,A_NORMAL);
        wgetch(win);
    }
    wclear(win);
    wresize(win,height-3,width);

    game_state = GAME_PLAYING;
}

void updatePlayer() {
    if (bankAccount >= 1000000000) retire();
    if (elapsedTime >= 2) renewLease();
}

void gameLoop() {
    while (quit == FALSE) {
        drawHud();
        drawPlay();
        drawGameMenu();
        getPlayerInput();
        updatePlayer();
    }
}

/*
    Initializes data and prepares the screen for display
*/
int initGame() {
    int c;

    srand (time(NULL));
#ifdef XCURSES
        Xinitscr(argc, argv);
#else
        initscr();
#endif
    raw();
    nodelay(stdscr,1);
    noecho();
    curs_set(0);
    nl();

#ifdef A_COLOR
        if (has_colors()) start_color();
#endif

    for (c = 0; c < COLORS; c++) {
        if (c == COLOR_WHITE) init_pair(c,COLOR_BLACK,c);
        else init_pair(c,COLOR_WHITE,c);
    }

    /* Special color pairs */
    init_pair(10,COLOR_YELLOW,COLOR_BLACK);
    init_pair(11,COLOR_GREEN,COLOR_BLACK);

       width  = 80;
       height = 25;

    clear();
    refresh();
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    /* Resize the terminal window */
    resize_term(height,width);

    win = newwin(height - 3, width, (LINES - height)/2, (COLS - width)/2);
    hud = newwin(3,width,height-3,0);
    manifest = newwin(height - 3, width, (LINES - height)/2, (COLS - width)/2);
    shipstatus = newwin(height - 3, width, (LINES - height)/2, (COLS - width)/2);

    infowin = newwin(3,40,10,20);

        if (win == NULL || hud == NULL ||
         manifest == NULL || infowin == NULL || shipstatus == NULL) {
                endwin();
                return 1;
        }

    savetty();
    resize_term(0,0);
    game_state = GAME_INTRO;
        return 0;
}

void gameShutdown() {
    endwin();
}

int main(int argc, char *argv[]) {

    setlocale(LC_ALL,"");

#ifdef PDCURSES
    PDC_set_title("Star Merchant");
#endif

        if (initGame())
                return 1;

    while (quit == FALSE) {
         switch(game_state) {
             case GAME_INTRO:
                 newGame();
                 break;
             case GAME_PLAYING:
                 gameLoop();
                 break;
             case GAME_DONE:
                 quit = TRUE;
                 break;
         }
    }

    gameShutdown();
        return 0;
}
```

/* Listing continued on next page…*/