Star Trek

The United Federation of Planets is under attack! A fleet of Klingon warships have invaded and it's up to you to stop them before they destroy Federation headquarters.

Star Trek simulates captaining a spaceship if that spaceship was built before gui interfaces. Part of the fun of the game is mastering its control scheme. You issue commands with three letter abbreviations and then input at the prompts. Mastering the computer (COM) commands can mean the difference between a great captain and a dead one. I may be helpful at first to keep a copy of the instructions nearby.

Game play usually goes like this: raise shields (SHE), long range scan (LRS) for Klingons, navigate (NAV) to the desired sector possibly using the computer (COM) direction/distance calculator (4) to help you chart the course, engage the Klingons using photon torpedoes when they're available (TOR) for a one hit kill or phasers (PHA) in a pinch, and refilling at star bases and repairing damage (DAM) whenever you need to.

If this game seems antiquated, that's because it is. Star Trek is an old game based on an old TV show. The original BASIC version was written in 1974. It has been written for almost every system and enhanced with graphics, sounds and gui interfaces. The version presented here is a conversion from the original BASIC version. It is no understatement to say that Star Trek the game holds a special place in a generation of gamer's hearts.

Star Trek is written by Chris Nystrom, converted from the BASIC game of the same name by Mike Mayfield, Robert Leedom and David H. Ahl as found in *101 BASIC Computer Games* edited by David H. Ahl, © 1984.

STARTREK.TXT

```
1. When you see _Command?_ printed, enter one of the legal commands
   (nav, srs, lrs, pha, tor, she, dam, com, or xxx).

2. If you should type in an illegal command, you'll get a short list of
   the legal commands printed out.

3. Some commands require you to enter data (for example, the 'nav' command
   comes back with 'Course(1-9) ?'.) If you type in illegal data (like
   negative numbers), that command will be aborted.

  The galaxy is divided into an 8 X 8 quadrant grid, and each quadrant
is further divided into an 8 x 8 sector grid.

  You will be assigned a starting point somewhere in the galaxy to begin
a tour of duty as commander of the starship _Enterprise_; your mission:
to seek out and destroy the fleet of Klingon warships which are menacing
the United Federation of Planets.

  You have the following commands available to you as Captain of the Starship
Enterprise:

\nav\ Command = Warp Engine Control --

  Course is in a circular numerical vector            4  3  2
  arrangement as shown. Integer and real               . . .
  values may be used. (Thus course 1.5 is               ...
  half-way between 1 and 2.                        5 ---*--- 1
                                                        ...
  Values may approach 9.0, which itself is             . . .
  equivalent to 1.0.                                   6  7  8

  One warp factor is the size of one quadrant.        COURSE
```

Listing continued on next page...

```
  Therefore, to get from quadrant 6,5 to 5,5
  you would use course 3, warp factor 1.

\srs\ Command = Short Range Sensor Scan

  Shows you a scan of your present quadrant.

  Symbology on your sensor screen is as follows:
    <*> = Your starship's position
    +K+ = Klingon battlecruiser
    >!< = Federation starbase (Refuel/Repair/Re-Arm here)
     *  = Star

  A condensed 'Status Report' will also be presented.

\lrs\ Command = Long Range Sensor Scan

  Shows conditions in space for one quadrant on each side of the Enterprise
  (which is in the middle of the scan). The scan is coded in the form \###\
  where the units digit is the number of stars, the tens digit is the number
  of starbases, and the hundreds digit is the number of Klingons.

  Example - 207 = 2 Klingons, No Starbases, & 7 stars.

\pha\ Command = Phaser Control.

  Allows you to destroy the Klingon Battle Cruisers by zapping them with
  suitably large units of energy to deplete their shield power. (Remember,
  Klingons have phasers, too!)

\tor\ Command = Photon Torpedo Control

  Torpedo course is the same  as used in warp engine control. If you hit
  the Klingon vessel, he is destroyed and cannot fire back at you. If you
  miss, you are subject to the phaser fire of all other Klingons in the
  quadrant.

  The Library-Computer (\com\ command) has an option to compute torpedo
  trajectory for you (option 2).

\she\ Command = Shield Control

  Defines the number of energy units to be assigned to the shields. Energy
  is taken from total ship's energy. Note that the status display total
  energy includes shield energy.

\dam\ Command = Damage Control report
  Gives the state of repair of all devices. Where a negative 'State of Repair'
  shows that the device is temporarily damaged.

\com\ Command = Library-Computer
  The Library-Computer contains six options:
  Option 0 = Cumulative Galactic Record
    This option shows computer memory of the results of all previous
    short and long range sensor scans.
  Option 1 = Status Report
    This option shows the number of Klingons, stardates, and starbases
    remaining in the game.
  Option 2 = Photon Torpedo Data
    Which gives directions and distance from Enterprise to all Klingons
```

Listing continued on next page...

STARTREK.TXT	Listing continued from previous page...

```
    in your quadrant.
  Option 3 = Starbase Nav Data
    This option gives direction and distance to any starbase in your
    quadrant.
  Option 4 = Direction/Distance Calculator
    This option allows you to enter coordinates for direction/distance
    calculations.
  Option 5 = Galactic /Region Name/ Map
    This option prints the names of the sixteen major galactic regions
    referred to in the game.
```

STARTREK.C	You will need: a C/C++ complier .

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

#ifndef FALSE
#define FALSE        0
#endif

#ifndef TRUE
#define TRUE         ! FALSE
#endif

/* Standard Line Length */

#define MAXLEN     255

/* Standard Terminal Sizes */

#define MAXROW      24
#define MAXCOL      80

/* Standard Page Size */

#define MAXLINES    66

/* Useful typedefs */

typedef char line[MAXCOL];
typedef char string[MAXLEN];

/* Function Declarations */

void intro(void);
void new_game(void);
void initialize(void);
void new_quadrant(void);
void course_control(void);
void complete_maneuver(void);
void exceed_quadrant_limits(void);
void maneuver_energy(void);
void short_range_scan(void);
void long_range_scan(void);
void phaser_control(void);
void photon_torpedoes(void);
void torpedo_hit(void);
```

Listing continued on next page...

```
void damage_control(void);
void sheild_control(void);
void library_computer(void);
void galactic_record(void);
void status_report(void);
void torpedo_data(void);
void nav_data(void);
void dirdist_calc(void);
void galaxy_map(void);
void end_of_time(void);
void resign_commision(void);
void won_game(void);
void end_of_game(void);
void klingons_move(void);
void klingons_shoot(void);
void repair_damage(void);
void find_empty_place(void);
void insert_in_quadrant(void);
void get_device_name(void);
void string_compare(void);
void quadrant_name(void);
int function_d(int i);
int function_r(void);
void mid_str(char *a, char *b, int x, int y);
int cint(double d);
void compute_vector(void);
void sub1(void);
void sub2(void);
void showfile(char *filename);
int openfile(char * sFilename, char * sMode);
void closefile(void);
int getline(char *s);
void randomize(void);
int get_rand(int iSpread);
double rnd(void);

/* Global Variables */

int b3;                          /* Starbases in Quadrant */
int b4, b5;                 /* Starbase Location in sector */
int b9;                             /* Total Starbases */

/* @@@ int c[2][10] = */  /* Used for location and movement */
int c[3][10] = { /* modified to match MS BASIC array indicies */
    { 0 },
    { 0, 0, -1, -1, -1,  0,  1, 1, 1, 0 },
    { 1, 1,  1,  0, -1, -1, -1, 0, 1, 1 }
};

int d0;                                    /* Docked flag */
int d1;                               /* Damage Repair Flag */
int e;                                 /* Current Energy */
int e0 = 3000;                         /* Starting Energy */
int g[9][9];                                    /* Galaxy */
int g5;                               /* Quadrant name flag */
int k[4][4];                                /* Klingon Data */
int k3;                              /* Klingons in Quadrant */
int k7;                                 /* Klingons at start */
int k9;                              /* Total Klingons left */
```

Listing continued on next page...

```
int n;                          /* Number of secors to travel */
int p;                            /* Photon Torpedoes left */
int p0 = 10;                     /* Photon Torpedo capacity */
int q1, q2;              /* Quadrant Position of Enterprise */
int r1, r2;               /* Temporary Location Corrdinates */
int s;                             /* Current shield value */
int s3;                               /* Stars in quadrant */
int s8;                          /* Quadrant locating index */
int s9 = 200;                               /* Klingon Power */
int t0;                                /* Starting Stardate */
int t9;                                     /* End of time */
int z[9][9];                /* Cumulative Record of Galaxy */
int z3;                     /* string_compare return value */
int z1, z2;                /* Temporary Sector Coordinates */
int z4, z5;               /* Temporary quadrant coordinates */

double a, c1;                    /* Used by Library Computer */
double d[9];                                 /* Damage Array */
double d4;          /* Used for computing damage repair time */
double s1, s2;      /* Current Sector Position of Enterprise */
double t;                                 /* Current Stardate */
double w1;                                     /* Warp Factor */
double x, y, x1, x2;            /* Navigational coordinates */

char sA[4];                       /* An Object in a Sector */
char sC[7];                                    /* Condition */
char sQ[194];                /* Visual Display of Quadrant */

string sG2;                 /* Used to pass string results */

FILE *stream;
int bFlag = FALSE;          /* Prevent multiple file opens */

/* Main Program */

int main(void) {
    intro();

    new_game();

    /* @@@ exit(0);  */ /* causes a warning in C++ */
    return(0);
}

void intro(void) {
    string sTemp;

    printf ("\n\n");
    printf (" ************************************\n");
    printf (" *                                  *\n");
    printf (" *                                  *\n");
    printf (" *      * * Super Star Trek * *     *\n");
    printf (" *                                  *\n");
    printf (" *                                  *\n");
    printf (" ************************************\n\n\n\n\n");

    printf("\nDo you need instructions (y/n): ");

    gets(sTemp);
```

Listing continued on next page...

```
    if (sTemp[0] == 'y' || sTemp[0] == 'Y')
        showfile("startrek.txt");

    printf ("\n\n\n\n\n\n\n\n");
    printf("                          ------*------\n");
    printf("         -------------   `---  ------'\n");
    printf("         `-------- --'      / /\n");
    printf("                  \\\\-------  --\n");
    printf("                  '-----------'\n");
    printf("\n       The USS Enterprise --- NCC - 1701\n\n\n");

    randomize();

    t = (get_rand(20) + 20) * 100;
}

void new_game(void) {
    string sTemp;

    initialize();

    new_quadrant();

    short_range_scan();

    while (1) {
        if (s + e <= 10 && (e < 10 || d[7] < 0)) {
            printf("\n** Fatal Error **   ");
            printf("You've just stranded your ship in space.\n\n");
            printf("You have insufficient maneuvering energy,");
            printf(" and Shield Control is presently\n");
            printf("incapable of cross circuiting to engine room!!\n\n");
            end_of_time();
        }

        printf("Command? ");

        gets(sTemp);
        printf("\n");

        if (! strncmp(sTemp, "nav", 3))
            course_control();
        else if (! strncmp(sTemp, "srs", 3))
            short_range_scan();
        else if (! strncmp(sTemp, "lrs", 3))
            long_range_scan();
        else if (! strncmp(sTemp, "pha", 3))
            phaser_control();
        else if (! strncmp(sTemp, "tor", 3))
            photon_torpedoes();
        else if (! strncmp(sTemp, "she", 3))
            sheild_control();
        else if (! strncmp(sTemp, "dam", 3))
            damage_control();
        else if (! strncmp(sTemp, "com", 3))
            library_computer();
        else if (! strncmp(sTemp, "xxx", 3))
            resign_commision();
```

Listing continued on next page...

```
        else {
            printf("Enter one of the following:\n\n");
            printf("  nav - To Set Course\n");
            printf("  srs - Short Range Sensors\n");
            printf("  lrs - Long Range Sensors\n");
            printf("  pha - Phasers\n");
            printf("  tor - Photon Torpedoes\n");
            printf("  she - Sheild Control\n");
            printf("  dam - Damage Control\n");
            printf("  com - Library Computer\n");
            printf("  xxx - Resign Command\n");
            printf("\n");
        }
    }
}

void initialize(void) {
    int i, j;
    char sX[2] = "";
    char sX0[4] = "is";

    /* InItialize time */

    /* @@@ t0 = t; */
    t0 = (int)t;
    t9 = 25 + get_rand(10);

    /* Initialize Enterprise */

    d0 = 0;
    e = e0;
    p = p0;
    s = 0;

    q1 = function_r();
    q2 = function_r();
    s1 = (double) function_r();
    s2 = (double) function_r();

    for (i = 1; i <= 8; i++)
        d[i] = 0.0;

    /* Setup What Exists in Galaxy */

    for (i = 1; i <= 8; i++)
        for (j = 1; j <= 8; j++) {
            k3 = 0;
            z[i][j] = 0;
            r1 = get_rand(100);
            if (r1 > 98)
                k3 = 3;
            else if (r1 > 95)
                k3 = 2;
            else if (r1 > 80)
                k3 = 1;

            k9 = k9 + k3;
            b3 = 0;
```

Listing continued on next page...

```
            if (get_rand(100) > 96)
                b3 = 1;

            b9 = b9 + b3;

            g[i][j] = k3 * 100 + b3 * 10 + function_r();
        }

    if (k9 > t9)
        t9 = k9 + 1;

    if (b9 == 0) {
        if (g[q1][q2] < 200) {
            g[q1][q2] = g[q1][q2] + 100;
            k9++;
        }

        g[q1][q2] = g[q1][q2] + 10;
        b9++;

        q1 = function_r();
        q2 = function_r();
    }

    k7 = k9;

    if (b9 != 1) {
        strcpy(sX, "s");
        strcpy(sX0, "are");
    }

    printf("Your orders are as follows:\n\n");
    printf("   Destroy the %d Klingon warships which have invaded\n", k9);
    printf(" the galaxy before they can attack Federation Headquarters\n");
    printf(" on stardate %d. This gives you %d days. There %s\n",
           t0 + t9, t9, sX0);
    printf(" %d starbase%s in the galaxy for resupplying your ship.\n\n",
           b9, sX);

    printf("Hit any key to accept command. ");
    getchar();
}

void new_quadrant(void) {
    int i;

    z4 = q1;
    z5 = q2;
    k3 = 0;
    b3 = 0;
    s3 = 0;
    g5 = 0;
    d4 = (double) get_rand(100) / 100 / 50;
    z[q1][q2] = g[q1][q2];

    if (q1 >= 1 && q1 <= 8 && q2 >= 1 && q2 <= 8) {
        quadrant_name();

        if (t0 != t)
```

Listing continued on next page...

```
            printf("Now entering %s quadrant...\n\n", sG2);
        else {
            printf("\nYour mission begins with your starship located\n");
            printf("in the galactic quadrant %s.\n\n", sG2);
        }
    }

    /* @@@ k3 = g[q1][q2] * .01; */
    k3 = (int)(g[q1][q2] * .01);
    /* @@@ b3 = g[q1][q2] * .1 - 10 * k3; */
    b3 = (int)(g[q1][q2] * .1 - 10 * k3);
    s3 = g[q1][q2] - 100 * k3 - 10 * b3;

    if (k3 > 0) {
        printf("Combat Area  Condition Red\n");

        if (s < 200)
            printf("Shields Dangerously Low\n");
    }

    for (i = 1; i <= 3; i++) {
        k[i][1] = 0;
        k[i][2] = 0;
        k[i][3] = 0;
    }

    for (i = 0; i <= 192; i++)
        sQ[i] = ' ';

    sQ[193] = '\0';

    /* Position Enterprise, then Klingons, Starbases, and stars */

    strcpy(sA, "<*>");
    /* @@@ z1 = cint(s1); */
    z1 = (int)s1;
    /* @@@ z2 = cint(s2); */
    z2 = (int)s2;
    insert_in_quadrant();

    if (k3 > 0) {
        for (i = 1; i <= k3; i++) {
            find_empty_place();

            strcpy(sA, "+K+");
            z1 = r1;
            z2 = r2;
            insert_in_quadrant();

            k[i][1] = r1;
            k[i][2] = r2;
            k[i][3] = 100 + get_rand(200);
        }
    }

    if (b3 > 0) {
        find_empty_place();

        strcpy(sA, ">!<");
```

Listing continued on next page...

```
        z1 = r1;
        z2 = r2;
        insert_in_quadrant();

        b4 = r1;
        b5 = r2;
    }

    for (i = 1; i <= s3; i++) {
        find_empty_place();

        strcpy(sA, " * ");
        z1 = r1;
        z2 = r2;
        insert_in_quadrant();
    }
}

void course_control(void) {
    int i;
    /* @@@ int c2, c3, q4, q5; */
    int q4, q5;
    string sTemp;
    double c1;
    char sX[4] = "8";

    printf("Course (0-9): ");

    gets(sTemp);

    printf("\n");

    c1 = atof(sTemp);

    if (c1 == 9.0)
        c1 = 1.0;

    if (c1 < 0 || c1 > 9.0) {
        printf("Lt. Sulu roports:\n");
        printf("  Incorrect course data, sir!\n\n");
        return;
    }

    if (d[1] < 0.0)
        strcpy(sX, "0.2");

    printf("Warp Factor (0-%s): ", sX);

    gets(sTemp);

    printf("\n");

    w1 = atof(sTemp);

    if (d[1] < 0.0 && w1 > 0.21) {
        printf("Warp Engines are damaged. ");
        printf("Maximum speed = Warp 0.2.\n\n");
        return;
    }
```

Listing continued on next page...

```
    if (w1 <= 0.0)
        return;

    if (w1 > 8.1) {
        printf("Chief Engineer Scott reports:\n");
        printf("  The engines won't take warp %4.1f!\n\n", w1);
        return;
    }

    n = cint(w1 * 8.0); /* @@@ note: this is a real round in the original basic
*/

    if (e - n < 0) {
        printf("Engineering reports:\n");
        printf("  Insufficient energy available for maneuvering");
        printf(" at warp %4.1f!\n\n", w1);

        if (s >= n && d[7] >= 0.0) {
            printf("Deflector Control Room acknowledges:\n");
            printf("  %d units of energy presently deployed to shields.\n", s);
        }

        return;
    }

    klingons_move();

    repair_damage();

    strcpy(sA, "   ");
    /* @@@ z1 = cint(s1); */
    z1 = (int)s1;
    /* @@@ z2 = cint(s2); */
    z2 = (int)s2;
    insert_in_quadrant();

    /* @@@ c2 = cint(c1); */
    /* @@@ c3 = c2 + 1; */

    /* @@@ x1 = c[0][c2] + (c[0][c3] - c[0][c2]) * (c1 - c2); */
    /* @@@ x2 = c[1][c2] + (c[1][c3] - c[1][c2]) * (c1 - c2); */

    x1 = c[1][(int)c1] + (c[1][(int)c1 + 1] - c[1][(int)c1]) * (c1 - (int)c1);
    x2 = c[2][(int)c1] + (c[2][(int)c1 + 1] - c[2][(int)c1]) * (c1 - (int)c1);

    x = s1;
    y = s2;
    q4 = q1;
    q5 = q2;

    for (i = 1; i <= n; i++) {
        s1 = s1 + x1;
        s2 = s2 + x2;

        /* @@@ z1 = cint(s1); */
        z1 = (int)s1;
        /* @@@ z2 = cint(s2); */
        z2 = (int)s2;
```

Listing continued on next page...

```
        if (z1 < 1 || z1 >= 9 || z2 < 1 || z2 >= 9) {
            exceed_quadrant_limits();
            complete_maneuver();
            return;
        }

        string_compare();

        if (z3 != 1) { /* Sector not empty */
            s1 = s1 - x1;
            s2 = s2 - x2;
            printf("Warp Engines shut down at sector ");
            printf("%d, %d due to bad navigation.\n\n", z1, z2);
            i = n + 1;
        }
    }

    complete_maneuver();
}

void complete_maneuver(void) {
    double t8;

    strcpy(sA, "<*>");
    /* @@@ z1 = cint(s1); */
    z1 = (int)s1;
    /* @@@ z2 = cint(s2); */
    z2 = (int)s2;
    insert_in_quadrant();

    maneuver_energy();

    t8 = 1.0;

    if (w1 < 1.0)
        t8 = w1;

    t = t + t8;

    if (t > t0 + t9)
        end_of_time();

    short_range_scan();
}

void exceed_quadrant_limits(void) {
    int x5 = 0;   /* Outside galaxy flag */

    /* @@@ x = (8 * (q1 - 1)) + x + (n * x1); */
    x = (8 * q1) + x + (n * x1);
    /* @@@ y = (8 * (q2 - 1)) + y + (n * x2); */
    y = (8 * q2) + y + (n * x2);

    /* @@@ q1 = cint(x / 8.0); */
    q1 = (int)(x / 8.0);
    /* @@@ q2 = cint(y / 8.0); */
    q2 = (int)(y / 8.0);
```

Listing continued on next page...

```
    /* @@@ s1 = x - ((q1 - 1) * 8); */
    s1 = x - (q1 * 8);
    /* @@@ s2 = y - ((q2 - 1) * 8); */
    s2 = y - (q2 * 8);

    /* @@@ if (cint(s1) == 0) */
    if ((int)s1 == 0) {
        q1 = q1 - 1;
        s1 = s1 + 8.0;
    }

    /* @@@ if (cint(s2) == 0) */
    if ((int)s2 == 0) {
        q2 = q2 - 1;
        s2 = s2 + 8.0;
    }

    /* check if outside galaxy */

    if (q1 < 1) {
        x5 = 1;
        q1 = 1;
        s1 = 1.0;
    }

    if (q1 > 8) {
        x5 = 1;
        q1 = 8;
        s1 = 8.0;
    }

    if (q2 < 1) {
        x5 = 1;
        q2 = 1;
        s2 = 1.0;
    }

    if (q2 > 8) {
        x5 = 1;
        q2 = 8;
        s2 = 8.0;
    }

    if (x5 == 1) {
        printf("LT. Uhura reports:\n");
        printf("  Message from Starfleet Command:\n\n");
        printf("  Permission to attempt crossing of galactic perimeter\n");
        printf("  is hereby *denied*. Shut down your engines.\n\n");
        printf("Chief Engineer Scott reports:\n");
        /* @@@ printf("  Warp Engines shut down at sector %d, ", cint(s1)); */
        printf("  Warp Engines shut down at sector %d, ", (int)s1);
        /* @@@ printf("%d of quadrant %d, %d.\n\n", cint(s2), q1, q2); */
        printf("%d of quadrant %d, %d.\n\n", (int)s2, q1, q2);
    }
    /* else
       new_quadrant(); @@@ this causes bugs when bouncing off galaxy walls.
                           basically, if you bounce very far, your quadrant con-
tents
                           won't match your LRS.  Cool huh? */
```

Listing continued on next page...

```
    maneuver_energy();

    /* this section has a different order in the original.
    t = t + 1;

    if (t > t0 + t9)
      end_of_time();
    */

    if (t > t0 + t9)
        end_of_time();

    /* @@@ what does this do?? It's in the original.
    if (8 * q1 + q2 = 8 * q4 + q5)
      {
        complete_maneuver();
      }
    */

    t = t + 1;

    new_quadrant();
}

void maneuver_energy(void) {
    e = e - n - 10;

    if (e >= 0)
        return;

    printf("Shield Control supplies energy to complete maneuver.\n\n");

    s = s + e;
    e = 0;

    if (s <= 0)
        s = 0;
}

void short_range_scan(void) {
    int i, j;

    strcpy(sC, "GREEN");

    if (e < e0 * .1)
        strcpy(sC, "YELLOW");

    if (k3 > 0)
        strcpy(sC, "*RED*");

    /* @@@ need to clear the docked flag here */
    d0 = 0;

    /* @@@ for (i = s1 - 1; i <= s1 + 1; i++) */
    for (i = (int)(s1 - 1); i <= (int)(s1 + 1); i++)
        /* @@@ for (j = s2 - 1; j <= s2 + 1; j++) */
        for (j = (int)(s2 - 1); j <= (int)(s2 + 1); j++)
```

Listing continued on next page...


```
            if (i >= 1 && i <= 8 && j >= 1 && j <= 8) {
                strcpy(sA, ">!<");
                z1 = i;
                z2 = j;
                string_compare();
                if (z3 == 1) {
                    d0 = 1;
                    strcpy(sC, "DOCKED");
                    e = e0;
                    p = p0;
                    printf("Shields dropped for docking purposes.\n");
                    s = 0;
                }
            }

    if (d[2] < 0.0) {
        printf("\n*** Short Range Sensors are out ***\n");
        return;
    }

    printf("-------------------------\n");
    for (i = 0; i < 8; i++) {
        for (j = 0; j < 24; j++)
            putchar(sQ[i * 24 + j]);

        if (i == 0)
            printf("    Stardate            %d\n", (int) t);
        if (i == 1)
            printf("    Condition           %s\n", sC);
        if (i == 2)
            printf("    Quadrant            %d, %d\n", q1, q2);
        if (i == 3)
            /* @@@ printf("    Sector              %d, %d\n", cint(s1), cint
(s2)); */
            printf("    Sector              %d, %d\n", (int)s1, (int)s2);
        if (i == 4)
            printf("    Photon Torpedoes    %d\n", p);
        if (i == 5)
            printf("    Total Energy        %d\n", e + s);
        if (i == 6)
            printf("    Shields             %d\n", s);
        if (i == 7)
            printf("    Klingons Remaining  %d\n", k9);
    }
    printf("-------------------------\n\n");

    return;
}

void long_range_scan(void) {
    int i, j;

    if (d[3] < 0.0) {
        printf("Long Range Sensors are inoperable.\n");
        return;
    }

    printf("Long Range Scan for Quadrant %d, %d\n\n", q1, q2);
```

Listing continued on next page...

```
    for (i = q1 - 1; i <= q1 + 1; i++) {
        printf("--------------------\n:");
        for (j = q2 - 1; j <= q2 + 1; j++)
            if (i > 0 && i <= 8 && j > 0 && j <= 8) {
                z[i][j] = g[i][j];
                printf(" %3.3d :", z[i][j]);
            } else
                printf(" *** :");
        printf("\n");
    }

    printf("--------------------\n\n");
}

void phaser_control(void) {
    int i;
    int iEnergy;
    int h1, h;
    string sTemp;

    if (d[4] < 0.0) {
        printf("Phasers Inoperative\n\n");
        return;
    }

    if (k3 <= 0) {
        printf("Science Officer Spock reports:\n");
        printf("  'Sensors show no enemy ships in this quadrant'\n\n");
        return;
    }

    if (d[8] < 0.0)
        /* @@@ printf("Computer failure happers accuracy.\n"); */
        printf("Computer failure hampers accuracy.\n");

    printf("Phasers locked on target;\n");
    printf("Energy available = %d units\n\n", e);

    printf("Number of units to fire: ");

    gets(sTemp);

    printf("\n");

    iEnergy = atoi(sTemp);

    if (iEnergy <= 0)
        return;

    if (e - iEnergy < 0) {
        printf("Not enough energy available.\n\n");
        return;
    }

    e = e - iEnergy;

    if (d[8] < 0.0)
        /* @@@ iEnergy = iEnergy * rnd(); */
        iEnergy = (int)(iEnergy * rnd());
```

Listing continued on next page...

```
    h1 = iEnergy / k3;

    for (i = 1; i <= 3; i++) {
        if (k[i][3] > 0) {
            /* @@@ h = (h1 / function_d(0) * (rnd() + 2)); */
            h = (int)(h1 / function_d(0) * (rnd() + 2));
            if (h <= .15 * k[i][3]) {
                printf("Sensors show no damage to enemy at ");
                printf("%d, %d\n\n", k[i][1], k[i][2]);
            } else {
                k[i][3] = k[i][3] - h;
                printf("%d unit hit on Klingon at sector ", h);
                printf("%d, %d\n", k[i][1], k[i][2]);
                if (k[i][3] <= 0) {
                    printf("*** Klingon Destroyed ***\n\n");
                    k3--;
                    k9--;
                    z1 = k[i][1];
                    z2 = k[i][2];
                    strcpy(sA, "   ");
                    insert_in_quadrant();
                    k[i][3] = 0;
                    g[q1][q2] = g[q1][q2] - 100;
                    z[q1][q2] = g[q1][q2];
                    if (k9 <= 0)
                        won_game();
                } else
                    /* @@@ printf("\n"); */
                    printf("   (Sensors show %d units remaining.)\n\n", k[i]
[3]);
            }
        }
    }

    klingons_shoot();
}

void photon_torpedoes(void) {
    /* @@@ int c2, c3, x3, y3, x5; */
    int x3, y3, x5;
    string sTemp;
    double c1;

    if (p <= 0) {
        printf("All photon torpedoes expended\n");
        return;
    }

    if (d[5] < 0.0) {
        printf("Photon Tubes not operational\n");
        return;
    }

    printf("Course (0-9): ");

    gets(sTemp);

    printf("\n");
```

Listing continued on next page...

```
    c1 = atof(sTemp);

    if (c1 == 9.0)
        c1 = 1.0;

    /* @@@ if (c1 < 0 || c1 > 9.0) */
    if (c1 < 1.0 || c1 > 9.0) {
        printf("Ensign Chekov roports:\n");
        printf("  Incorrect course data, sir!\n\n");
        return;
    }

    e = e - 2;
    p--;

    /* @@@ c2 = cint(c1); */
    /* @@@ c3 = c2 + 1; */

    /* @@@ x1 = c[0][c2] + (c[0][c3] - c[0][c2]) * (c1 - c2); */
    /* @@@ x2 = c[1][c2] + (c[1][c3] - c[1][c2]) * (c1 - c2); */

    x1 = c[1][(int)c1] + (c[1][(int)c1 + 1] - c[1][(int)c1]) * (c1 - (int)c1);
    x2 = c[2][(int)c1] + (c[2][(int)c1 + 1] - c[2][(int)c1]) * (c1 - (int)c1);

    x = s1 + x1;
    y = s2 + x2;

    x3 = cint(x); /* @@@ note: this is a true integer round in the MS BASIC ver-
sion */
    y3 = cint(y); /* @@@ note: this is a true integer round in the MS BASIC ver-
sion */

    x5 = 0;

    printf("Torpedo Track:\n");

    while (x3 >= 1 && x3 <= 8 && y3 >= 1 && y3 <= 8) {
        printf("    %d, %d\n", x3, y3);

        strcpy(sA, "   ");
        z1 = x3;
        z2 = y3;

        string_compare();

        if (z3 == 0) {
            torpedo_hit();
            klingons_shoot();
            return;
        }

        x = x + x1;
        y = y + x2;

        x3 = cint(x); /* @@@ note: this is a true integer round in the MS BASIC
version */
        y3 = cint(y); /* @@@ note: this is a true integer round in the MS BASIC
version */
```

Listing continued on next page...

```
    }

    printf("Torpedo Missed\n\n");

    klingons_shoot();
}

void torpedo_hit(void) {
    int i, x3, y3;

    x3 = cint(x); /* @@@ note: this is a true integer round in the MS BASIC ver-
sion */
    y3 = cint(y); /* @@@ note: this is a true integer round in the MS BASIC ver-
sion */

    z3 = 0;

    strcpy(sA, " * ");
    string_compare();

    if (z3 == 1) {
        printf("Star at %d, %d absorbed torpedo energy.\n\n", x3, y3);
        return;
    }

    strcpy(sA, "+K+");
    string_compare();

    if (z3 == 1) {
        printf("*** Klingon Destroyed ***\n\n");
        k3--;
        k9--;

        if (k9 <= 0)
            won_game();

        for (i=0; i<=3; i++)
            if (x3 == k[i][1] && y3 == k[i][2])
                k[i][3] = 0;
    }

    strcpy(sA, ">!<");
    string_compare();

    if (z3 == 1) {
        printf("*** Starbase Destroyed ***\n");
        b3--;
        b9--;

        if (b9 <= 0 && k9 <= t - t0 - t9) {
            printf("That does it, Captain!!");
            printf("You are hereby relieved of command\n");
            printf("and sentanced to 99 stardates of hard");
            printf("labor on Cygnus 12!!\n");
            resign_commision();
        }

        printf("Starfleet Command reviewing your record to consider\n");
        printf("court martial!\n\n");
```

Listing continued on next page...


```
        d0 = 0;    /* Undock */
    }

    z1 = x3;
    z2 = y3;
    strcpy(sA,"   ");
    insert_in_quadrant();

    g[q1][q2] = (k3 * 100) + (b3 * 10) + s3;
    z[q1][q2] = g[q1][q2];
}

void damage_control(void) {
    int a1;
    double d3 = 0.0;
    int i;

    if (d[6] < 0.0) {
        printf("Damage Control report not available.\n");

        if (d0 == 0)
            return;

        d3 = 0.0;
        for (i = 1; i <= 8; i++)
            if (d[i] < 0.0)
                d3 = d3 + .1;

        if (d3 == 0.0)
            return;

        d3 = d3 + d4;
        if (d3 >= 1.0)
            d3 = 0.9;

        printf("\nTechnicians standing by to effect repairs to your");
        /* @@@ printf("ship; Will you authorize the repair order (Y/N)? "); */
        printf("ship;\nEstimated time to repair: %4.2f stardates.\n", d3);
        printf("Will you authorize the repair order (Y/N)? ");

        a1 = getchar();

        if (a1 == 'Y' || a1 == 'y') {
            for (i = 1; i <= 8; i++)
                if (d[i] < 0.0)
                    d[i] = 0.0;

            t = t + d3 + 0.1;
        }
    }

    printf("Device            State of Repair\n");

    for (r1 = 1; r1 <= 8; r1++) {
        get_device_name();
        printf(sG2);
        /* @@@ for (i = 1; i < 25 - strlen(sG2); i++) */
        for (i = 1; i < 25 - (int)strlen(sG2); i++)
```

Listing continued on next page...

```
            printf(" ");
        /* @@@ printf("%4.1f\n", d[r1]); */
        printf("%4.2f\n", d[r1]);
    }

    printf("\n");
}

void sheild_control(void) {
    int i;
    string sTemp;

    if (d[7] < 0.0) {
        printf("Sheild Control inoperable\n");
        return;
    }

    printf("Energy available = %d\n\n", e + s);

    printf("Input number of units to shields: ");

    gets(sTemp);

    printf("\n");

    i = atoi(sTemp);

    if (i < 0 || s == i) {
        printf("<Sheilds Unchanged>\n\n");
        return;
    }

    if (i >= e + s) {
        printf("Sheild Control Reports:\n");
        printf("  'This is not the Federation Treasury.'\n");
        printf("<Sheilds Unchanged>\n\n");
        return;
    }

    e = e + s - i;
    s = i;

    printf("Deflector Control Room report:\n");
    printf("  'Shields now at %d units per your command.'\n\n", s);
}

void library_computer(void) {
    string sTemp;

    if (d[8] < 0.0) {
        printf("Library Computer inoperable\n");
        return;
    }

    printf("Computer active and awating command: ");

    gets(sTemp);
    printf("\n");
```

Listing continued on next page...

```
    if (! strncmp(sTemp, "0", 1))
        galactic_record();
    else if (! strncmp(sTemp, "1", 1))
        status_report();
    else if (! strncmp(sTemp, "2", 1))
        torpedo_data();
    else if (! strncmp(sTemp, "3", 1))
        nav_data();
    else if (! strncmp(sTemp, "4", 1))
        dirdist_calc();
    else if (! strncmp(sTemp, "5", 1))
        galaxy_map();
    else {
        printf("Functions available from Library-Computer:\n\n");
        printf("   0 = Cumulative Galactic Record\n");
        printf("   1 = Status Report\n");
        printf("   2 = Photon Torpedo Data\n");
        printf("   3 = Starbase Nav Data\n");
        printf("   4 = Direction/Distance Calculator\n");
        printf("   5 = Galaxy 'Region Name' Map\n\n");
    }
}

void galactic_record(void) {
    int i, j;

    printf("\n     Computer Record of Galaxy for Quadrant %d,%d\n\n", q1, q2);
    printf("     1     2     3     4     5     6     7     8\n");

    for (i = 1; i <= 8; i++) {
        printf("   ----- ----- ----- ----- ----- ----- ----- -----\n");

        printf("%d", i);

        for (j = 1; j <= 8; j++) {
            printf("   ");

            if (z[i][j] == 0)
                printf("***");
            else
                printf("%3.3d", z[i][j]);
        }

        printf("\n");
    }

    printf("   ----- ----- ----- ----- ----- ----- ----- -----\n\n");
}

void status_report(void) {
    char sX[2] = "";

    printf("   Status Report:\n\n");

    if (k9 > 1)
        strcpy(sX, "s");

    printf("Klingon%s Left: %d\n", sX, k9);
```

Listing continued on next page...

```
    printf("Mission must be completed in %4.1f stardates\n",
           /* @@@ .1 * cint((t0 + t9 - t) * 10)); */
           .1 * (int)((t0 + t9 - t) * 10));

    if (b9 < 1) {
        printf("Your stupidity has left you on your own in the galaxy\n");
        printf(" -- you have no starbases left!\n");
    } else {
        strcpy(sX, "s");
        if (b9 < 2)
            strcpy(sX, "");

        printf("The Federation is maintaining %d starbase%s in the galaxy\n",
               b9, sX);
    }

    printf("\n");
}

void torpedo_data(void) {
    int i;
    char sX[2] = "";

    if (k3 <= 0) {
        printf("Science Officer Spock reports:\n");
        printf("  'Sensors show no enemy ships in this quadrant.'\n\n");
        return;
    }

    if (k3 > 1)
        strcpy(sX, "s");

    printf("From Enterprise to Klingon battlecriuser%s:\n\n", sX);

    for (i = 1; i <= 3; i++) {
        if (k[i][3] > 0) {
            w1 = k[i][1];
            x  = k[i][2];
            c1 = s1;
            a  = s2;

            compute_vector();
        }
    }
}

void nav_data(void) {
    if (b3 <= 0) {
        printf("Mr. Spock reports,\n");
        printf("  'Sensors show no starbases in this quadrant.'\n\n");
        return;
    }

    w1 = b4;
    x  = b5;
    c1 = s1;
    a  = s2;

    compute_vector();
```

Listing continued on next page...

```
}

void dirdist_calc(void) {
    string sTemp;

    printf("Direction/Distance Calculator\n\n");
    printf("You are at quadrant %d,%d sector %d,%d\n\n", q1, q2,
           /* @@@ cint(s1), cint(s2)); */
           (int)s1, (int)s2);

    printf("Please enter initial X coordinate: ");
    gets(sTemp);
    c1 = atoi(sTemp);

    printf("Please enter initial Y coordinate: ");
    gets(sTemp);
    a = atoi(sTemp);

    printf("Please enter final X coordinate: ");
    gets(sTemp);
    w1 = atoi(sTemp);

    printf("Please enter final Y coordinate: ");
    gets(sTemp);
    x = atoi(sTemp);

    compute_vector();
}

void galaxy_map(void) {
    int i, j, j0;

    g5 = 1;

    printf("\n                    The Galaxy\n\n");
    printf("    1     2     3     4     5     6     7     8\n");

    for (i = 1; i <= 8; i++) {
        printf("  ----- ----- ----- ----- ----- ----- ----- -----\n");

        printf("%d ", i);

        z4 = i;
        z5 = 1;
        quadrant_name();

        j0 = (int)(11 - (strlen(sG2) / 2));

        for (j = 0; j < j0; j++)
            printf(" ");

        printf(sG2);

        for (j = 0; j < j0; j++)
            printf(" ");

        if (! (strlen(sG2) % 2))
            printf(" ");
```

Listing continued on next page...

```
        z5 = 5;
        quadrant_name();

        j0 = (int)(12 - (strlen(sG2) / 2));

        for (j = 0; j < j0; j++)
            printf(" ");

        printf(sG2);

        printf("\n");
    }

    printf("  ----- ----- ----- ----- ----- ----- ----- -----\n\n");

}

void compute_vector(void) {
    x = x - a;
    a = c1 - w1;

    if (x <= 0.0) {
        if (a > 0.0) {
            c1 = 3.0;
            sub2();
            return;
        } else {
            c1 = 5.0;
            sub1();
            return;
        }
    } else if (a < 0.0) {
        c1 = 7.0;
        sub2();
        return;
    } else {
        c1 = 1.0;
        sub1();
        return;
    }
}

void sub1(void) {
    x = fabs(x);
    a = fabs(a);

    if (a <= x)
        printf("  DIRECTION = %4.2f\n", c1 + (a / x));
    else
        printf("  DIRECTION = %4.2f\n", c1 + (((a * 2) - x) / a));

    printf("  DISTANCE = %4.2f\n\n", (x > a) ? x : a);
}

void sub2(void) {
    x = fabs(x);
    a = fabs(a);

    if (a >= x)
```

Listing continued on next page...

```
        printf("  DIRECTION = %4.2f\n", c1 + (x / a));
    else
        /* @@@ printf("  DIRECTION = %4.2f\n\n", c1 + (((x * 2) - a) / x)); */
        printf("  DIRECTION = %4.2f\n", c1 + (((x * 2) - a) / x));

    /* @@@ printf("  DISTANCE = %4.2f\n", (x > a) ? x : a); */
    printf("  DISTANCE = %4.2f\n\n", (x > a) ? x : a);
}

void ship_destroyed(void) {
    printf("The Enterprise has been destroyed. ");
    printf("The Federation will be conquered.\n\n");

    end_of_time();
}

void end_of_time(void) {
    printf("It is stardate %d.\n\n", (int) t);

    resign_commision();
}

void resign_commision(void) {
    printf("There were %d Klingon Battlecruisers left at the", k9);
    printf(" end of your mission.\n\n");

    end_of_game();
}

void won_game(void) {
    printf("Congradulations, Captain!  The last Klingon Battle Cruiser\n");
    printf("menacing the Federation has been destoyed.\n\n");

    if (t - t0 > 0)
        printf("Your efficiency rating is %4.2f\n", 1000 * pow(k7 / (t - t0),
2));

    end_of_game();
}

void end_of_game(void) {
    string sTemp;

    if (b9 > 0) {
        printf("The Federation is in need of a new starship commander");
        printf(" for a similar mission.\n");
        printf("If there is a volunteer, let him step forward and");
        printf(" enter 'aye': ");

        gets(sTemp);
        printf("\n");

        if (! strncmp(sTemp, "aye", 3))
            new_game();
    }

    exit(0);
}
```

Listing continued on next page...

```
void klingons_move(void) {
    int i;

    for (i = 1; i <= 3; i++) {
        if (k[i][3] > 0) {
            strcpy(sA, "   ");
            z1 = k[i][1];
            z2 = k[i][2];
            insert_in_quadrant();

            find_empty_place();

            k[i][1] = z1;
            k[i][2] = z2;
            strcpy(sA, "+K+");
            insert_in_quadrant();
        }
    }

    klingons_shoot();
}

void klingons_shoot(void) {
    int h, i;

    if (k3 <= 0)
        return;

    if (d0 != 0) {
        printf("Starbase shields protect the Enterprise\n\n");
        return;
    }

    for (i = 1; i <= 3; i++) {
        if (k[i][3] > 0) {
            h = (int) ((k[i][3] / function_d(i)) * (2 + rnd()));
            s = s - h;
            /* @@@ k[i][3] = k[i][3] / (3 + rnd()); */
            k[i][3] = (int)(k[i][3] / (3 + rnd()));

            printf("%d unit hit on Enterprise from sector ", h);
            printf("%d, %d\n", k[i][1], k[i][2]);

            if (s <= 0) {
                printf("\n");
                ship_destroyed();
            }

            printf("    <Shields down to %d units>\n\n", s);

            if (h >= 20) {
                if (rnd() <= 0.6 || (h / s) > 0.2) {
                    r1 = function_r();
                    d[r1] = d[r1] - (h / s) - (0.5 * rnd());

                    get_device_name();

                    printf("Damage Control reports\n");
                    printf("   '%s' damaged by hit\n\n", sG2);
```

Listing continued on next page...

```
                }
            }
        }
    }
}

void repair_damage(void) {
    int i;
    double d6;              /* Repair Factor */

    d6 = w1;

    if (w1 >= 1.0)
        d6 = w1 / 10;

    for (i = 1; i <= 8; i++) {
        if (d[i] < 0.0) {
            d[i] = d[i] + d6;
            if (d[i] > -0.1 && d[i] < 0)
                d[i] = -0.1;
            else if (d[i] >= 0.0) {
                if (d1 != 1)
                    d1 = 1;

                printf("Damage Control report:\n");
                r1 = i;
                get_device_name();
                printf("    %s repair completed\n\n", sG2);
            }
        }
    }

    if (rnd() <= 0.2) {
        r1 = function_r();

        if (rnd() < .6) {
            d[r1] = d[r1] - (rnd() * 5.0 + 1.0);
            printf("Damage Control report:\n");
            get_device_name();
            printf("    %s damaged\n\n", sG2);
        } else {
            d[r1] = d[r1] + (rnd() * 3.0 + 1.0);
            printf("Damage Control report:\n");
            get_device_name();
            printf("    %s state of repair improved\n\n", sG2);
        }
    }
}

/* Misc Functions and Subroutines */

void find_empty_place(void) {
    /* @@@ while (z3 == 0) this is a nasty one.*/
    do {
        r1 = function_r();
        r2 = function_r();

        strcpy(sA, "   ");
```

Listing continued on next page...

```
        z1 = r1;
        z2 = r2;

        string_compare();
    } while (z3 == 0);

    z3 = 0;
}

void insert_in_quadrant(void) {
    int i, j = 0;

    /* @@@ s8 = ((z2 - 1) * 3) + ((z1 - 1) * 24) + 1; */
    s8 = ((int)(z2 - 0.5) * 3) + ((int)(z1 - 0.5) * 24) + 1;

    for (i = s8 - 1; i <= s8 + 1; i++)
        sQ[i] = sA[j++];

    return;
}

void get_device_name(void) {
    static char * device_name[] = {
        "", "Warp Engines","Short Range Sensors","Long Range Sensors",
        "Phaser Control","Photon Tubes","Damage Control","Sheild Control",
        "Library-Computer"
    };

    if (r1 < 0 || r1 > 8)
        r1 = 0;

    strcpy(sG2, device_name[r1]);

    return;
}

void string_compare(void) {
    int i;
    char sB[4];

    z1 = (int)(z1 + 0.5);
    z2 = (int)(z2 + 0.5);

    s8 = ((z2 - 1) * 3) + ((z1 - 1) * 24) + 1;

    mid_str(sB, sQ, s8, 3);

    i = strncmp(sB, sA, 3);

    if (i == 0)
        z3 = 1;
    else
        z3 = 0;

    return;
}

void quadrant_name(void) {
    static char * quad_name[] = {"","Antares","Rigel","Procyon","Vega",
```

Listing continued on next page...

```
"Canopus","Altair","Sagittarius","Pollux","Sirius","Deneb","Capella",

"Betelgeuse","Aldebaran","Regulus","Arcturus","Spica"
                                };

    static char * sect_name[] = {"","," I","," II","," III","," IV"};

    if (z4 < 1 || z4 > 8 || z5 < 1 || z5 > 8)
        strcpy(sG2, "Unknown");

    if (z5 <= 4)
        strcpy(sG2, quad_name[z4]);
    else
        strcpy(sG2, quad_name[z4+8]);

    if (g5 != 1) {
        if (z5 > 4)
            z5 = z5 - 4;
        strcat(sG2, sect_name[z5]);
    }

    return;
}

int
function_d(int i) {
    int j;

    /* @@@ j = sqrt(pow((k[i][1] - s1), 2) + pow((k[i][2] - s2), 2)); */
    j = (int)sqrt(pow((k[i][1] - s1), 2) + pow((k[i][2] - s2), 2));

    return j;
}

int function_r(void) {
    return(get_rand(8));
}

void mid_str(char *a, char *b, int x, int y) {
    --x;
    y += x;

    /* @@@ while (x < y && x <= strlen(b)) */
    while (x < y && x <= (int)strlen(b))
        *a++ = *(b + x++);

    *a = '\0';
}

/* Round off floating point numbers instead of truncating */

int cint (double d) {
    int i;

    i = (int) (d + 0.5);

    return(i);
}
```

Listing continued on next page...