```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)getpar.h    8.1 (Berkeley) 5/31/93
 * $DragonFly: src/games/trek/getpar.h,v 1.2 2006/09/07 21:19:44 pavalos Exp $
 */

#include        <stdbool.h>

struct cvntab           /* used for getcodpar() parameter list */
{
        const char      *abrev;
        const char      *full;
        void    (*value)(int);
        int     value2;
};

extern struct cvntab    Lentab[];
extern struct cvntab    Skitab[];

int     getintpar(const char *);
double  getfltpar(const char *);
long    getynpar(const char *);
struct cvntab   *getcodpar(const char *, struct cvntab *);
void    getstrpar(const char *, char *, int, const char *);
bool    testnl(void);
void    skiptonl(char);
bool    readdelim(char);
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 *      @(#)trek.h      8.1 (Berkeley) 5/31/93
 * $DragonFly: src/games/trek/trek.h,v 1.2 2006/09/07 21:19:44 pavalos Exp $
 */

#include        <math.h>
#include        <setjmp.h>
#include        <stdbool.h>
#include        <stdio.h>
#include        <stdlib.h>
#include        <string.h>
#include        <unistd.h>
/*
**  Global Declarations
**
**      Virtually all non-local variable declarations are made in this
**      file.  Exceptions are those things which are initialized, which
**      are defined in "externs.c", and things which are local to one
**      program file.
**
**      So far as I know, nothing in here must be preinitialized to
**      zero.
**
**      You may have problems from the loader if you move this to a
**      different machine.  These things actually get allocated in each
**      source file, which UNIX allows; however, you may (on other
**      systems) have to change everything in here to be "extern" and
**      actually allocate stuff in "externs.c"
*/

extern  jmp_buf env;
```

```
/*********************  GALAXY  *************************/

/* galactic parameters */
# define        NSECTS          10      /* dimensions of quadrant in sectors */
# define        NQUADS          8       /* dimension of galazy in quadrants */
# define        NINHAB          32      /* number of quadrants which are inhabit
ed */

struct quad             /* definition for each quadrant */
{
        char    bases;          /* number of bases in this quadrant */
        char    klings;         /* number of Klingons in this quadrant */
        char    holes;          /* number of black holes in this quadrant */
        int     scanned;        /* star chart entry (see below) */
        char    stars;          /* number of stars in this quadrant */
        char    qsystemname;    /* starsystem name (see below) */
};

# define        Q_DISTRESSED    0200
# define        Q_SYSTEM        077

/*  systemname conventions:
 *      1 -> NINHAB     index into Systemname table for live system.
 *      + Q_DISTRESSED  distressed starsystem -- systemname & Q_SYSTEM
 *                      is the index into the Event table which will
 *                      have the system name
 *      0               dead or nonexistent starsystem
 *
 *  starchart ("scanned") conventions:
 *      0 -> 999        taken as is
 *      -1              not yet scanned ("...")
 *      1000            supernova ("///")
 *      1001            starbase + ??? (".1.")
*/

/* ascii names of systems */
extern const char       *Systemname[NINHAB];

/* quadrant definition */
struct quad     Quad[NQUADS][NQUADS];

/* defines for sector map  (below) */
# define        EMPTY           '.'
# define        STAR            '*'
# define        BASE            '#'
# define        ENTERPRISE      'E'
# define        QUEENE          'Q'
# define        KLINGON         'K'
# define        INHABIT         '@'
# define        HOLE            ' '

/* current sector map */
char    Sect[NSECTS][NSECTS];


/*********************** DEVICES ****************************/

# define        NDEV            16      /* max number of devices */

/* device tokens */
# define        WARP            0       /* warp engines */
# define        SRSCAN          1       /* short range scanners */
# define        LRSCAN          2       /* long range scanners */
# define        PHASER          3       /* phaser control */
# define        TORPED          4       /* photon torpedo control */
# define        IMPULSE         5       /* impulse engines */
# define        SHIELD          6       /* shield control */
# define        COMPUTER        7       /* on board computer */
# define        SSRADIO         8       /* subspace radio */
# define        LIFESUP         9       /* life support systems */
# define        SINS            10      /* Space Inertial Navigation System */
# define        CLOAK           11      /* cloaking device */
# define        XPORTER         12      /* transporter */
# define        SHUTTLE         13      /* shuttlecraft */

/* device names */
struct device
{
        const char      *name;          /* device name */
        const char      *person;        /* the person who fixes it */
};

extern struct device    Device[NDEV];

/***************************  EVENTS  ****************************/

# define        NEVENTS         12      /* number of different event types */

# define        E_LRTB          1       /* long range tractor beam */
# define        E_KATSB         2       /* Klingon attacks starbase */
# define        E_KDESB         3       /* Klingon destroys starbase */
# define        E_ISSUE         4       /* distress call is issued */
# define        E_ENSLV         5       /* Klingons enslave a quadrant */
# define        E_REPRO         6       /* a Klingon is reproduced */
# define        E_FIXDV         7       /* fix a device */
# define        E_ATTACK        8       /* Klingon attack during rest period */
# define        E_SNAP          9       /* take a snapshot for time warp */
# define        E_SNOVA         10      /* supernova occurs */

# define        E_GHOST         0100    /* ghost of a distress call if ssradio o
ut */
# define        E_HIDDEN        0200    /* event that is unreportable because ss
radio out */
# define        E_EVENT         077     /* mask to get event code */

struct event
{
        short   x, y;                   /* coordinates */
        double  date;                   /* trap stardate */
        char    evcode;                 /* event type */
        short   systemname;             /* starsystem name */
};
/* systemname conventions:
 *      1 -> NINHAB     index into Systemname table for reported distress calls
 *
 * evcode conventions:
 *      1 -> NEVENTS-1  event type
 *      + E_HIDDEN      unreported (SSradio out)
 *      + E_GHOST       actually already expired
 *      0               unallocated
 */

# define        MAXEVENTS       25      /* max number of concurrently pending ev
ents */
```

```

struct event    Event[MAXEVENTS];       /* dynamic event list; one entry per pen
ding event */

/****************************  KLINGONS  ******************************/

struct kling
{
        short   x, y;           /* coordinates */
        int     power;          /* power left */
        double  dist;           /* distance to Enterprise */
        double  avgdist;        /* average over this move */
        char    srndreq;        /* set if surrender has been requested */
};

# define        MAXKLQUAD       9       /* maximum klingons per quadrant */

/********************** MISCELLANEOUS *************************/

/* condition codes */
# define        GREEN           0
# define        DOCKED          1
# define        YELLOW          2
# define        RED             3

/* starbase coordinates */
# define        MAXBASES        9       /* maximum number of starbases in galaxy
 */

/*  distress calls  */
# define        MAXDISTR        5       /* maximum concurrent distress calls */

/* phaser banks */
# define        NBANKS          6       /* number of phaser banks */

struct xy
{
        short   x, y;           /* coordinates */
};


/*
 *      note that much of the stuff in the following structs CAN NOT
 *      be moved around!!!!
 */


/* information regarding the state of the starship */
struct
{
        double  warp;           /* warp factor */
        double  warp2;          /* warp factor squared */
        double  warp3;          /* warp factor cubed */
        char    shldup;         /* shield up flag */
        char    cloaked;        /* set if cloaking device on */
        int     energy;         /* starship's energy */
        int     shield;         /* energy in shields */
        double  reserves;       /* life support reserves */
        int     crew;           /* ship's complement */
        int     brigfree;       /* space left in brig */
        char    torped;         /* torpedoes */
        char    cloakgood;      /* set if we have moved */
        int     quadx;          /* quadrant x coord */
        int     quady;          /* quadrant y coord */
        int     sectx;          /* sector x coord */
        int     secty;          /* sector y coord */
        short   cond;           /* condition code */
        char    sinsbad;        /* Space Inertial Navigation System condition */
        const char      *shipname;      /* name of current starship */
        char    ship;           /* current starship */
        int     distressed;     /* number of distress calls */
}       Ship;

/* sinsbad is set if SINS is working but not calibrated */

/* game related information, mostly scoring */
struct
{
        int     killk;          /* number of klingons killed */
        int     deaths;         /* number of deaths onboard Enterprise */
        char    negenbar;       /* number of hits on negative energy barrier */
        char    killb;          /* number of starbases killed */
        int     kills;          /* number of stars killed */
        char    skill;          /* skill rating of player */
        char    length;         /* length of game */
        char    killed;         /* set if you were killed */
        char    killinhab;      /* number of inhabited starsystems killed */
        char    tourn;          /* set if a tournament game */
        char    passwd[15];     /* game password */
        char    snap;           /* set if snapshot taken */
        char    helps;          /* number of help calls */
        int     captives;       /* total number of captives taken */
}       Game;

/* per move information */
struct
{
        char    free;           /* set if a move is free */
        char    endgame;        /* end of game flag */
        char    shldchg;        /* set if shields changed this move */
        char    newquad;        /* set if just entered this quadrant */
        char    resting;        /* set if this move is a rest */
        double  time;           /* time used this move */
}       Move;

/* parametric information */
struct
{
        char    bases;          /* number of starbases */
        char    klings;         /* number of klingons */
        double  date;           /* stardate */
        double  time;           /* time left */
        double  resource;       /* Federation resources */
        int     energy;         /* starship's energy */
        int     shield;         /* energy in shields */
        double  reserves;       /* life support reserves */
        int     crew;           /* size of ship's complement */
        int     brigfree;       /* max possible number of captives */
        char    torped;         /* photon torpedos */
        double  damfac[NDEV];   /* damage factor */
        double  dockfac;        /* docked repair time factor */
        double  regenfac;       /* regeneration factor */
        int     stopengy;       /* energy to do emergency stop */
        int     shupengy;       /* energy to put up shields */
```

```
        int     klingpwr;       /* Klingon initial power */
        int     warptime;       /* time chewer multiplier */
        double  phasfac;        /* Klingon phaser power eater factor */
        char    moveprob[6];    /* probability that a Klingon moves */
        double  movefac[6];     /* Klingon move distance multiplier */
        double  eventdly[NEVENTS];      /* event time multipliers */
        double  navigcrud[2];   /* navigation crudup factor */
        int     cloakenergy;    /* cloaking device energy per stardate */
        double  damprob[NDEV];  /* damage probability */
        double  hitfac;         /* Klingon attack factor */
        int     klingcrew;      /* number of Klingons in a crew */
        double  srndrprob;      /* surrender probability */
        int     energylow;      /* low energy mark (cond YELLOW) */
}       Param;

/* Sum of damage probabilities must add to 1000 */

/* other information kept in a snapshot */
struct
{
        short   bases;          /* number of starbases */
        char    klings;         /* number of klingons */
        double  date;           /* stardate */
        double  time;           /* time left */
        double  resource;       /* Federation resources */
        char    distressed;     /* number of currently distressed quadrants */
        struct event    *eventptr[NEVENTS];     /* pointer to event structs */
        struct xy       base[MAXBASES];         /* locations of starbases */
}       Now;

/* Other stuff, not dumped in a snapshot */
struct
{
        struct kling    klingon[MAXKLQUAD];     /* sorted Klingon list */
        int             nkling;                 /* number of Klingons in this se
ctor */
                                                /* < 0 means automatic override
mode */
        struct xy       starbase;       /* starbase in current quadrant */
        char            snapshot[sizeof Quad + sizeof Event + sizeof Now];
/* snapshot for time warp */
        char            statreport;             /* set to get a status report on
 a srscan */
}       Etc;

/*
 *      eventptr is a pointer to the event[] entry of the last
 *      scheduled event of each type.  Zero if no such event scheduled.
 */

/* Klingon move indicies */
# define        KM_OB           0       /* Old quadrant, Before attack */
# define        KM_OA           1       /* Old quadrant, After attack */
# define        KM_EB           2       /* Enter quadrant, Before attack */
# define        KM_EA           3       /* Enter quadrant, After attack */
# define        KM_LB           4       /* Leave quadrant, Before attack */
# define        KM_LA           5       /* Leave quadrant, After attack */

/* you lose codes */
# define        L_NOTIME        1       /* ran out of time */
# define        L_NOENGY        2       /* ran out of energy */
# define        L_DSTRYD        3       /* destroyed by a Klingon */
# define        L_NEGENB        4       /* ran into the negative energy barrier
*/
# define        L_SUICID        5       /* destroyed in a nova */
# define        L_SNOVA         6       /* destroyed in a supernova */
# define        L_NOLIFE        7       /* life support died (so did you) */
# define        L_NOHELP        8       /* you could not be rematerialized */
# define        L_TOOFAST       9       /* pretty stupid going at warp 10 */
# define        L_STAR          10      /* ran into a star */
# define        L_DSTRCT        11      /* self destructed */
# define        L_CAPTURED      12      /* captured by Klingons */
# define        L_NOCREW        13      /* you ran out of crew */

/******************  COMPILE OPTIONS  ***********************/

/* Trace info */
# define        xTRACE          1
int     Trace;

/* external function definitions */
void    abandon(int);
void    attack(int);
void    autover(void);
void    capture(int);
int     cgetc(int);
bool    check_out(int);
void    checkcond(void);
void    compkldist(bool);
void    computer(int);
void    damage(int, double);
bool    damaged(int);
void    dcrept(int);
void    destruct(int);
void    dock(int);
void    undock(int);
void    dumpgame(int);
bool    restartgame(void);
void    dumpme(int);
int     dumpssradio(void);
void    events(int);
bool    getcodi(int *, double *);
void    help(int);
void    impulse(int);
void    initquad(int);
void    sector(int *, int *);
void    killk(int, int);
void    killb(int, int);
void    kills(int, int, int);
void    killd(int, int, int);
void    klmove(int);
void    lose(int);
void    lrscan(int);
double  move(int, int, double, double);
void    nova(int, int);
void    out(int);
void    phaser(int);
void    play(void);
void    ram(int, int);
int     ranf(int);
double  franf(void);
void    rest(int);
struct event    *schedule(int, double, char, char, char);
void    reschedule(struct event *, double);
```

```
void    unschedule(struct event *);
struct event    *xsched(int, int, int, int, int);
void    xresched(struct event *, int, int);
long    score(void);
void    setup(void);
void    setwarp(int);
void    shield(int);
void    snova(int, int);
void    srscan(int);
const char      *systemname(struct quad *);
void    torped(int);
char    *bmove(const void *, void *, size_t);
bool    sequal(const char *, const char *);
void    syserr(const char *, ...);
void    visual(int);
void    warp(int, int, double);
void    dowarp(int);
void    win(void);
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)abandon.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/abandon.c,v 1.4 1999/11/30 03:49:43 billf Exp $
 * $DragonFly: src/games/trek/abandon.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Abandon Ship
**
**      The ship is abandoned.  If your current ship is the Faire
**      Queene, or if your shuttlecraft is dead, you’re out of
**      luck.  You need the shuttlecraft in order for the captain
**      (that’s you!!) to escape.
**
**      Your crew can beam to an inhabited starsystem in the
**      quadrant, if there is one and if the transporter is working.
**      If there is no inhabited starsystem, or if the transporter
**      is out, they are left to die in outer space.
**
**      These currently just count as regular deaths, but they
**      should count very heavily against you.
**
**      If there are no starbases left, you are captured by the
**      Klingons, who torture you mercilessly.  However, if there
**      is at least one starbase, you are returned to the
**      Federation in a prisoner of war exchange.  Of course, this
**      can’t happen unless you have taken some prisoners.
**
**      Uses trace flag 40
```

```
*/

void
abandon(__unused int unused)
{
        struct quad     *q;
        int             i;
        int                     j;
        struct event    *e;

        if (Ship.ship == QUEENE) {
                printf("You may not abandon ye Faire Queene\n");
                return;
        }
        if (Ship.cond != DOCKED)
        {
                if (damaged(SHUTTLE)) {
                        out(SHUTTLE);
                        return;
                }
                printf("Officers escape in shuttlecraft\n");
                /* decide on fate of crew */
                q = &Quad[Ship.quadx][Ship.quady];
                if (q->qsystemname == 0 || damaged(XPORTER))
                {
                        printf("Entire crew of %d left to die in outer space\n",
                                Ship.crew);
                        Game.deaths += Ship.crew;
                }
                else
                {
                        printf("Crew beams down to planet %s\n", systemname(q));
                }
        }
        /* see if you can be exchanged */
        if (Now.bases == 0 || Game.captives < 20 * Game.skill)
                lose(L_CAPTURED);
        /* re-outfit new ship */
        printf("You are hereby put in charge of an antiquated but still\n");
        printf(" functional ship, the Fairie Queene.\n");
        Ship.ship = QUEENE;
        Ship.shipname = "Fairie Queene";
        Param.energy = Ship.energy = 3000;
        Param.torped = Ship.torped = 6;
        Param.shield = Ship.shield = 1250;
        Ship.shldup = 0;
        Ship.cloaked = 0;
        Ship.warp = 5.0;
        Ship.warp2 = 25.0;
        Ship.warp3 = 125.0;
        Ship.cond = GREEN;
        /* clear out damages on old ship */
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode != E_FIXDV)
                        continue;
                unschedule(e);
        }
        /* get rid of some devices and redistribute probabilities */
        i = Param.damprob[SHUTTLE] + Param.damprob[CLOAK];
        Param.damprob[SHUTTLE] = Param.damprob[CLOAK] = 0;
        while (i > 0)
                for (j = 0; j < NDEV; j++)
                {
                        if (Param.damprob[j] != 0)
                        {
                                Param.damprob[j] += 1;
                                i--;
                                if (i <= 0)
                                        break;
                        }
                }
        /* pick a starbase to restart at */
        i = ranf(Now.bases);
        Ship.quadx = Now.base[i].x;
        Ship.quady = Now.base[i].y;
        /* setup that quadrant */
        while (1)
        {
                initquad(1);
                Sect[Ship.sectx][Ship.secty] = EMPTY;
                for (i = 0; i < 5; i++)
                {
                        Ship.sectx = Etc.starbase.x + ranf(3) - 1;
                        if (Ship.sectx < 0 || Ship.sectx >= NSECTS)
                                continue;
                        Ship.secty = Etc.starbase.y + ranf(3) - 1;
                        if (Ship.secty < 0 || Ship.secty >= NSECTS)
                                continue;
                        if (Sect[Ship.sectx][Ship.secty] == EMPTY)
                        {
                                Sect[Ship.sectx][Ship.secty] = QUEENE;
                                dock(0);
                                compkldist(0);
                                return;
                        }
                }
        }
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)attack.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/attack.c,v 1.4 1999/11/30 03:49:43 billf Exp $
 * $DragonFly: src/games/trek/attack.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Klingon Attack Routine
**
**      This routine performs the Klingon attack provided that
**      (1) Something happened this move (i.e., not free), and
**      (2) You are not cloaked.  Note that if you issue the
**      cloak command, you are not considered cloaked until you
**      expend some time.
**
**      Klingons are permitted to move both before and after the
**      attack.  They will tend to move toward you before the
**      attack and away from you after the attack.
**
**      Under certain conditions you can get a critical hit.  This
**      sort of hit damages devices.  The probability that a given
**      device is damaged depends on the device.  Well protected
**      devices (such as the computer, which is in the core of the
**      ship and has considerable redundancy) almost never get
**      damaged, whereas devices which are exposed (such as the
**      warp engines) or which are particularly delicate (such as
**      the transporter) have a much higher probability of being
**      damaged.
**
```

```
**      The actual amount of damage (i.e., how long it takes to fix
**      it) depends on the amount of the hit and the "damfac[]"
**      entry for the particular device.
**
**      Casualties can also occur.
*/

void
attack(int resting)
/* resting:  set if attack while resting */
{
        int             hit, i, l;
        int                     maxhit, tothit, shldabsb;
        double                  chgfac, propor, extradm;
        double                  dustfac, tothe;
        int                     cas;
        int                     hitflag;

        if (Move.free)
                return;
        if (Etc.nkling <= 0 || Quad[Ship.quadx][Ship.quady].stars < 0)
                return;
        if (Ship.cloaked && Ship.cloakgood)
                return;
        /* move before attack */
        klmove(0);
        if (Ship.cond == DOCKED)
        {
                if (!resting)
                        printf("Starbase shields protect the %s\n", Ship.shipname);
                return;
        }
        /* setup shield effectiveness */
        chgfac = 1.0;
        if (Move.shldchg)
                chgfac = 0.25 + 0.50 * franf();
        maxhit = tothit = 0;
        hitflag = 0;

        /* let each Klingon do his damndest */
        for (i = 0; i < Etc.nkling; i++)
        {
                /* if he’s low on power he won’t attack */
                if (Etc.klingon[i].power < 20)
                        continue;
                if (!hitflag)
                {
                        printf("\nStardate %.2f: Klingon attack:\n",
                                Now.date);
                        hitflag++;
                }
                /* complete the hit */
                dustfac = 0.90 + 0.01 * franf();
                tothe = Etc.klingon[i].avgdist;
                hit = Etc.klingon[i].power * pow(dustfac, tothe) * Param.hitfac;
                /* deplete his energy */
                dustfac = Etc.klingon[i].power;
                Etc.klingon[i].power = dustfac * Param.phasfac * (1.0 + (franf()
 − 0.5) * 0.2);
                /* see how much of hit shields will absorb */
                shldabsb = 0;
                if (Ship.shldup || Move.shldchg)
                {
                        propor = Ship.shield;
                        propor /= Param.shield;
                        shldabsb = propor * chgfac * hit;
                        if (shldabsb > Ship.shield)
                                shldabsb = Ship.shield;
                        Ship.shield -= shldabsb;
                }
                /* actually do the hit */
                printf("^GHIT: %d units", hit);
                if (!damaged(SRSCAN))
                        printf(" from %d,%d", Etc.klingon[i].x, Etc.klingon[i].y);
                cas = (shldabsb * 100) / hit;
                hit -= shldabsb;
                if (shldabsb > 0)
                        printf(", shields absorb %d%%, effective hit %d\n",
                                cas, hit);
                else
                        printf("\n");
                tothit += hit;
                if (hit > maxhit)
                        maxhit = hit;
                Ship.energy -= hit;
                /* see if damages occurred */
                if (hit >= (15 - Game.skill) * (25 - ranf(12)))
                {
                        printf("^GCRITICAL HIT!!!^G\n");
                        /* select a device from probability vector */
                        cas = ranf(1000);
                        for (l = 0; cas >= 0; l++)
                                cas -= Param.damprob[l];
                        l -= 1;
                        /* compute amount of damage */
                        extradm = (hit * Param.damfac[l]) / (75 + ranf(25)) + 0.
5;
                        /* damage the device */
                        damage(l, extradm);
                        if (damaged(SHIELD))
                        {
                                if (Ship.shldup)
                                        printf("Sulu: Shields knocked down, captain.\n");
                                Ship.shldup = 0;
                                Move.shldchg = 0;
                        }
                }
                if (Ship.energy <= 0)
                        lose(L_DSTRYD);
        }

        /* see what our casualities are like */
        if (maxhit >= 200 || tothit >= 500)
        {
                cas = tothit * 0.015 * franf();
                if (cas >= 2)
                {
                        printf("McCoy: we suffered %d casualties in that attack.\n",
                                cas);
                        Game.deaths += cas;
                        Ship.crew -= cas;
                }
        }
```

```
        /* allow Klingons to move after attacking */
        klmove(1);

        return;
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)autover.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/autover.c,v 1.4 1999/11/30 03:49:43 billf Exp $
 * $DragonFly: src/games/trek/autover.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Automatic Override
**
**      If we should be so unlucky as to be caught in a quadrant
**      with a supernova in it, this routine is called.  It is
**      called from checkcond().
**
**      It sets you to a random warp (guaranteed to be over 6.0)
**      and starts sending you off "somewhere" (whereever that is).
**
**      Please note that it is VERY important that you reset your
**      warp speed after the automatic override is called.  The new
**      warp factor does not stay in effect for just this routine.
**
**      This routine will never try to send you more than sqrt(2)
**      quadrants, since that is all that is needed.
*/

void
autover(void)
{
        double                  dist;
        int             course;
```

```c

        printf("\07RED ALERT:  The %s is in a supernova quadrant\n", Ship.shipname);
        printf("***  Emergency override attempts to hurl %s to safety\n", Ship.shipname);
        /* let's get our ass out of here */
        Ship.warp = 6.0 + 2.0 * franf();
        Ship.warp2 = Ship.warp * Ship.warp;
        Ship.warp3 = Ship.warp2 * Ship.warp;
        dist = 0.75 * Ship.energy / (Ship.warp3 * (Ship.shldup + 1));
        if (dist > 1.4142)
                dist = 1.4142;
        course = ranf(360);
        Etc.nkling = -1;
        Ship.cond = RED;
        warp(-1, course, dist);
        attack(0);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)capture.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/capture.c,v 1.4 1999/11/30 03:49:43 billf Exp $
 * $DragonFly: src/games/trek/capture.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

static struct kling     *selectklingon(void);

/*
**  Ask a Klingon To Surrender
**
**      (Fat chance)
**
**      The Subspace Radio is needed to ask a Klingon if he will kindly
**      surrender.  A random Klingon from the ones in the quadrant is
**      chosen.
**
**      The Klingon is requested to surrender.  The probability of this
**      is a function of that Klingon's remaining power, our power,
**      etc.
*/

void
capture(__unused int unused)
{
        int             i;
        struct kling    *k;
        double                  x;
```

```c
        /* check for not cloaked */
        if (Ship.cloaked)
        {
                printf("Ship−ship communications out when cloaked\n");
                return;
        }
        if (damaged(SSRADIO))
                return (out(SSRADIO));
        /* find out if there are any at all */
        if (Etc.nkling <= 0)
        {
                printf("Uhura: Getting no response, sir\n");
                return;
        }

        /* if there is more than one Klingon, find out which one */
        k = selectklingon();
        Move.free = 0;
        Move.time = 0.05;

        /* check out that Klingon */
        k−>srndreq++;
        x = Param.klingpwr;
        x *= Ship.energy;
        x /= k−>power * Etc.nkling;
        x *= Param.srndrprob;
        i = x;
#       ifdef xTRACE
        if (Trace)
                printf("Prob = %d (%.4f)\n", i, x);
#       endif
        if (i > ranf(100))
        {
                /* guess what, he surrendered!!! */
                printf("Klingon at %d,%d surrenders\n", k−>x, k−>y);
                i = ranf(Param.klingcrew);
                if ( i > 0 )
                        printf("%d klingons commit suicide rather than be taken captive\n", Param
.klingcrew − i);
                if (i > Ship.brigfree)
                        i = Ship.brigfree;
                Ship.brigfree −= i;
                printf("%d captives taken\n", i);
                killk(k−>x, k−>y);
                return;
        }

        /* big surprise, he refuses to surrender */
        printf("Fat chance, captain\n");
        return;
}


/*
**  SELECT A KLINGON
**
**      Cruddy, just takes one at random.  Should ask the captain.
*/

static struct kling *
selectklingon(void)
{
        int             i;

        if (Etc.nkling < 2)
                i = 0;
        else
                i = ranf(Etc.nkling);
        return (&Etc.klingon[i]);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)cgetc.c  8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/cgetc.c,v 1.2 1999/11/30 03:49:44 billf Exp $
 * $DragonFly: src/games/trek/cgetc.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

int
cgetc(__unused int i)
{
        return ( getchar() );
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)check_out.c      8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/check_out.c,v 1.4 1999/11/30 03:49:44 billf Exp $
 * $DragonFly: src/games/trek/check_out.c,v 1.3 2006/09/07 21:19:44 pavalos Exp
$
 */

# include       "trek.h"

/*
**  CHECK IF A DEVICE IS OUT
**
**      The indicated device is checked to see if it is disabled.  If
**      it is, an attempt is made to use the starbase device.  If both
**      of these fails, it returns non-zero (device is REALLY out),
**      otherwise it returns zero (I can get to it somehow).
**
**      It prints appropriate messages too.
*/

bool
check_out(int device)
{
        int     dev;

        dev = device;

        /* check for device ok */
        if (!damaged(dev))
                return (0);
```

```
        /* report it as being dead */
        out(dev);

        /* but if we are docked, we can go ahead anyhow */
        if (Ship.cond != DOCKED)
                return (1);
        printf(" Using starbase %s\n", Device[dev].name);
        return (0);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)checkcond.c      8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/checkcond.c,v 1.4 1999/11/30 03:49:44 billf Exp $
 * $DragonFly: src/games/trek/checkcond.c,v 1.3 2006/09/07 21:19:44 pavalos Exp
$
 */

# include       "trek.h"

/*
**  Check for Condition After a Move
**
**      Various ship conditions are checked.  First we check
**      to see if we have already lost the game, due to running
**      out of life support reserves, running out of energy,
**      or running out of crew members.  The check for running
**      out of time is in events().
**
**      If we are in automatic override mode (Etc.nkling < 0), we
**      don’t want to do anything else, lest we call autover
**      recursively.
**
**      In the normal case, if there is a supernova, we call
**      autover() to help us escape.  If after calling autover()
**      we are still in the grips of a supernova, we get burnt
**      up.
**
**      If there are no Klingons in this quadrant, we nullify any
**      distress calls which might exist.
**
**      We then set the condition code, based on the energy level
```

```c
**      and battle conditions.
*/

void
checkcond(void)
{
        /* see if we are still alive and well */
        if (Ship.reserves < 0.0)
                lose(L_NOLIFE);
        if (Ship.energy <= 0)
                lose(L_NOENGY);
        if (Ship.crew <= 0)
                lose(L_NOCREW);
        /* if in auto override mode, ignore the rest */
        if (Etc.nkling < 0)
                return;
        /* call in automatic override if appropriate */
        if (Quad[Ship.quadx][Ship.quady].stars < 0)
                autover();
        if (Quad[Ship.quadx][Ship.quady].stars < 0)
                lose(L_SNOVA);
        /* nullify distress call if appropriate */
        if (Etc.nkling <= 0)
                killd(Ship.quadx, Ship.quady, 1);

        /* set condition code */
        if (Ship.cond == DOCKED)
                return;

        if (Etc.nkling > 0)
        {
                Ship.cond = RED;
                return;
        }
        if (Ship.energy < Param.energylow)
        {
                Ship.cond = YELLOW;
                return;
        }
        Ship.cond = GREEN;
        return;
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)compkl.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/compkl.c,v 1.4 1999/11/30 03:49:44 billf Exp $
 * $DragonFly: src/games/trek/compkl.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

static void     sortkl(void);

/*
**  compute klingon distances
**
**      The klingon list has the distances for all klingons recomputed
**      and sorted.  The parameter is a Boolean flag which is set if
**      we have just entered a new quadrant.
**
**      This routine is used every time the Enterprise or the Klingons
**      move.
*/

void
compkldist(bool f)
{
        int             i, dx, dy;
        double                  d;
        double                  temp;

        if (Etc.nkling == 0)
                return;
        for (i = 0; i < Etc.nkling; i++)
```

```c
        {
                /* compute distance to the Klingon */
                dx = Ship.sectx − Etc.klingon[i].x;
                dy = Ship.secty − Etc.klingon[i].y;
                d = dx * dx + dy * dy;
                d = sqrt(d);

                /* compute average of new and old distances to Klingon */
                if (!f)
                {
                        temp = Etc.klingon[i].dist;
                        Etc.klingon[i].avgdist = 0.5 * (temp + d);
                }
                else
                {
                        /* new quadrant: average is current */
                        Etc.klingon[i].avgdist = d;
                }
                Etc.klingon[i].dist = d;
        }

        /* leave them sorted */
        sortkl();
}


/*
**  sort klingons
**
**      bubble sort on ascending distance
*/

static void
sortkl(void)
{
        struct kling            t;
        int             f, i, m;

        m = Etc.nkling − 1;
        f = 1;
        while (f)
        {
                f = 0;
                for (i = 0; i < m; i++)
                        if (Etc.klingon[i].dist > Etc.klingon[i+1].dist)
                        {
                                bmove(&Etc.klingon[i], &t, sizeof t);
                                bmove(&Etc.klingon[i+1], &Etc.klingon[i], sizeof
 t);
                                bmove(&t, &Etc.klingon[i+1], sizeof t);
                                f = 1;
                        }
        }
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)computer.c       8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/computer.c,v 1.5 1999/11/30 03:49:45 billf Exp $
 * $DragonFly: src/games/trek/computer.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

static int      kalc(int, int, int, int, double *);
static void     prkalc(int, double);

/*
**  On-Board Computer
**
**      A computer request is fetched from the captain.  The requests
**      are:
**
**      chart -- print a star chart of the known galaxy.  This includes
**              every quadrant that has ever had a long range or
**              a short range scan done of it, plus the location of
**              all starbases.  This is of course updated by any sub-
**              space radio broadcasts (unless the radio is out).
**              The format is the same as that of a long range scan
**              except that ".1." indicates that a starbase exists
**              but we know nothing else.
**
**      trajectory -- gives the course and distance to every know
**              Klingon in the quadrant.  Obviously this fails if the
**              short range scanners are out.
**
```

```
**      course -- gives a course computation from whereever you are
**              to any specified location.  If the course begins
**              with a slash, the current quadrant is taken.
**              Otherwise the input is quadrant and sector coordi-
**              nates of the target sector.
**
**      move -- identical to course, except that the move is performed.
**
**      score -- prints out the current score.
**
**      pheff -- "PHaser EFFectiveness" at a given distance.  Tells
**              you how much stuff you need to make it work.
**
**      warpcost -- Gives you the cost in time and units to move for
**              a given distance under a given warp speed.
**
**      impcost -- Same for the impulse engines.
**
**      distresslist -- Gives a list of the currently known starsystems
**              or starbases which are distressed, together with their
**              quadrant coordinates.
**
**      If a command is terminated with a semicolon, you remain in
**      the computer; otherwise, you escape immediately to the main
**      command processor.
*/

struct cvntab   Cputab[] =
{
        { "ch", "art",                  (void (*)(int))1,       0 },
        { "t",  "rajectory",            (void (*)(int))2,       0 },
        { "c",  "ourse",                (void (*)(int))3,       0 },
        { "m",  "ove",                  (void (*)(int))3,       1 },
        { "s",  "core",                 (void (*)(int))4,       0 },
        { "p",  "heff",                 (void (*)(int))5,       0 },
        { "w",  "arpcost",              (void (*)(int))6,       0 },
        { "i",  "mpcost",               (void (*)(int))7,       0 },
        { "d",  "istresslist",          (void (*)(int))8,       0 },
        { NULL, NULL,                   NULL,                   0 }
};

void
computer(__unused int unused)
{
        int                     ix, iy;
        int             i, j;
        int                     tqx, tqy;
        struct cvntab           *r;
        int                     cost;
        int                     course;
        double                  dist, p_time;
        double                  warpfact;
        struct quad             *q;
        struct event    *e;

        if (check_out(COMPUTER))
                return;
        while (1)
        {
                r = getcodpar("\nRequest", Cputab);
                switch ((long)r->value)
                {
```

```
                case 1:                         /* star chart */
                        printf("Computer record of galaxy for all long range sensor scans\n\n");
                        printf(" ");
                        /* print top header */
                        for (i = 0; i < NQUADS; i++)
                                printf("−%d− ", i);
                        printf("\n");
                        for (i = 0; i < NQUADS; i++)
                        {
                                printf("%d ", i);
                                for (j = 0; j < NQUADS; j++)
                                {
                                        if (i == Ship.quadx && j == Ship.quady)
                                        {
                                                printf("$$$ ");
                                                continue;
                                        }
                                        q = &Quad[i][j];
                                        /* 1000 or 1001 is special case */
                                        if (q−>scanned >= 1000)
                                                if (q−>scanned > 1000)
                                                        printf(".1. ");
                                                else
                                                        printf("/// ");
                                        else
                                                if (q−>scanned < 0)
                                                        printf("... ");
                                                else
                                                        printf("%3d ", q−>scanne
d);
                                }
                                printf("%d\n", i);
                        }
                        printf(" ");
                        /* print bottom footer */
                        for (i = 0; i < NQUADS; i++)
                                printf("−%d− ", i);
                        printf("\n");
                        break;

                case 2:                         /* trajectory */
                        if (check_out(SRSCAN))
                        {
                                break;
                        }
                        if (Etc.nkling <= 0)
                        {
                                printf("No Klingons in this quadrant\n");
                                break;
                        }
                        /* for each Klingon, give the course & distance */
                        for (i = 0; i < Etc.nkling; i++)
                        {
                                printf("Klingon at %d,%d", Etc.klingon[i].x, Etc.kl
ingon[i].y);
                                course = kalc(Ship.quadx, Ship.quady, Etc.klingo
n[i].x, Etc.klingon[i].y, &dist);
                                prkalc(course, dist);
                        }
                        break;
```

```
                case 3:                         /* course calculation */
                        if (readdelim('/'))
                        {
                                tqx = Ship.quadx;
                                tqy = Ship.quady;
                        }
                        else
                        {
                                ix = getintpar("Quadrant");
                                if (ix < 0 || ix >= NSECTS)
                                        break;
                                iy = getintpar("q−y");
                                if (iy < 0 || iy >= NSECTS)
                                        break;
                                tqx = ix;
                                tqy = iy;
                        }
                        ix = getintpar("Sector");
                        if (ix < 0 || ix >= NSECTS)
                                break;
                        iy = getintpar("s−y");
                        if (iy < 0 || iy >= NSECTS)
                                break;
                        course = kalc(tqx, tqy, ix, iy, &dist);
                        if (r−>value2)
                        {
                                warp(−1, course, dist);
                                break;
                        }
                        printf("%d,%d/%d,%d to %d,%d/%d,%d",
                                Ship.quadx, Ship.quady, Ship.sectx, Ship.secty,
tqx, tqy, ix, iy);
                        prkalc(course, dist);
                        break;

                case 4:                         /* score */
                        score();
                        break;

                case 5:                         /* phaser effectiveness */
                        dist = getfltpar("range");
                        if (dist < 0.0)
                                break;
                        dist *= 10.0;
                        cost = pow(0.90, dist) * 98.0 + 0.5;
                        printf("Phasers are %d%% effective at that range\n", cost);
                        break;

                case 6:                         /* warp cost (time/energy) */
                        dist = getfltpar("distance");
                        if (dist < 0.0)
                                break;
                        warpfact = getfltpar("warp factor");
                        if (warpfact <= 0.0)
                                warpfact = Ship.warp;
                        cost = (dist + 0.05) * warpfact * warpfact * warpfact;
                        p_time = Param.warptime * dist / (warpfact * warpfact);
                        printf("Warp %.2f distance %.2f cost %.2f stardates %d (%d w/ shlds up) units
\n",
                                warpfact, dist, p_time, cost, cost + cost);
                        break;
                case 7:                         /* impulse cost */
                        dist = getfltpar("distance");
                        if (dist < 0.0)
                                break;
                        cost = 20 + 100 * dist;
                        p_time = dist / 0.095;
                        printf("Distance %.2f cost %.2f stardates %d units\n",
                                dist, p_time, cost);
                        break;

                case 8:                         /* distresslist */
                        j = 1;
                        printf("\n");
                        /* scan the event list */
                        for (i = 0; i < MAXEVENTS; i++)
                        {
                                e = &Event[i];
                                /* ignore hidden entries */
                                if (e->evcode & E_HIDDEN)
                                        continue;
                                switch (e->evcode & E_EVENT)
                                {

                                  case E_KDESB:
                                        printf("Klingon is attacking starbase in quadrant %d,%
d\n",
                                                e->x, e->y);
                                        j = 0;
                                        break;

                                  case E_ENSLV:
                                  case E_REPRO:
                                        printf("Starsystem %s in quadrant %d,%d is distressed\
n",
                                                Systemname[e->systemname], e->x,
 e->y);
                                        j = 0;
                                        break;
                                }
                        }
                        if (j)
                                printf("No known distress calls are active\n");
                        break;

                }

                /* skip to next semicolon or newline.  Semicolon
                 * means get new computer request; newline means
                 * exit computer mode. */
                while ((i = cgetc(0)) != ';')
                {
                        if (i == '\0')
                                exit(1);
                        if (i == '\n')
                        {
                                ungetc(i, stdin);
                                return;
                        }
                }
        }
}
```

```

/*
**  Course Calculation
**
**      Computes and outputs the course and distance from position
**      sqx,sqy/ssx,ssy to tqx,tqy/tsx,tsy.
*/

static int
kalc(int tqx, int tqy, int tsx, int tsy, double *dist)
{
        double                  dx, dy;
        double                  quadsize;
        double                  angle;
        int             course;

        /* normalize to quadrant distances */
        quadsize = NSECTS;
        dx = (Ship.quadx + Ship.sectx / quadsize) − (tqx + tsx / quadsize);
        dy = (tqy + tsy / quadsize) − (Ship.quady + Ship.secty / quadsize);

        /* get the angle */
        angle = atan2(dy, dx);
        /* make it 0 −> 2 pi */
        if (angle < 0.0)
                angle += 6.283185307;
        /* convert from radians to degrees */
        course = angle * 57.29577951 + 0.5;
        dx = dx * dx + dy * dy;
        *dist = sqrt(dx);
        return (course);
}

static void
prkalc(int course, double dist)
{
        printf(": course %d  dist %.3f\n", course, dist);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)damage.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/damage.c,v 1.4 1999/11/30 03:49:45 billf Exp $
 * $DragonFly: src/games/trek/damage.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Schedule Ship.damages to a Device
**
**      Device `dev1' is damaged in an amount `dam'.  Dam is measured
**      in stardates, and is an additional amount of damage.  It should
**      be the amount to occur in non-docked mode.  The adjustment
**      to docked mode occurs automatically if we are docked.
**
**      Note that the repair of the device occurs on a DATE, meaning
**      that the dock() and undock() have to reschedule the event.
*/

void
damage(int dev1, double dam)
{
        int             i;
        struct event    *e;
        int                     f;
        int             dev;

        /* ignore zero damages */
        if (dam <= 0.0)
                return;
```

```
        dev = dev1;

        printf("\t%s damaged\n", Device[dev].name);

        /* find actual length till it will be fixed */
        if (Ship.cond == DOCKED)
                dam *= Param.dockfac;
        /* set the damage flag */
        f = damaged(dev);
        if (!f)
        {
                /* new damages -- schedule a fix */
                schedule(E_FIXDV, dam, 0, 0, dev);
                return;
        }
        /* device already damaged -- add to existing damages */
        /* scan for old damages */
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode != E_FIXDV || e->systemname != dev)
                        continue;
                /* got the right one; add on the new damages */
                reschedule(e, e->date - Now.date + dam);
                return;
        }
        syserr("Cannot find old damages %d\n", dev);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)damaged.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/damaged.c,v 1.4 1999/11/30 03:49:45 billf Exp $
 * $DragonFly: src/games/trek/damaged.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include        "trek.h"

/*  DAMAGED -- check for device damaged
**
**      This is a boolean function which returns non-zero if the
**      specified device is broken.  It does this by checking the
**      event list for a "device fix" action on that device.
*/

bool
damaged(int dev)
{
        int             d;
        struct event    *e;
        int             i;

        d = dev;

        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode != E_FIXDV)
                        continue;
                if (e->systemname == d)
                        return (1);
```

```
        }

        /* device fix not in event list -- device must not be broken */
        return (0);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)dcrept.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/dcrept.c,v 1.4 1999/11/30 03:49:46 billf Exp $
 * $DragonFly: src/games/trek/dcrept.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  damage control report
**
**      Print damages and time to fix.  This is taken from the event
**      list.  A couple of factors are set up, based on whether or not
**      we are docked.  (One of these factors will always be 1.0.)
**      The event list is then scanned for damage fix events, the
**      time until they occur is determined, and printed out.  The
**      magic number DAMFAC is used to tell how much faster you can
**      fix things if you are docked.
*/

void
dcrept(__unused int unused)
{
        int             i, f;
        double                  x;
        double                  m1, m2;
        struct event    *e;

        /* set up the magic factors to output the time till fixed */
        if (Ship.cond == DOCKED)
        {
```

```
                m1 = 1.0 / Param.dockfac;
                m2 = 1.0;
        }
        else
        {
                m1 = 1.0;
                m2 = Param.dockfac;
        }
        printf("Damage control report:\n");
        f = 1;

        /* scan for damages */
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode != E_FIXDV)
                        continue;

                /* output the title first time */
                if (f)
                {
                        printf("\t\t\t  repair times\n");
                        printf("device\t\t\tin flight  docked\n");
                        f = 0;
                }

                /* compute time till fixed, then adjust by the magic factors */
                x = e->date − Now.date;
                printf("%−24s%7.2f  %7.2f\n",
                        Device[e->systemname].name, x * m1 + 0.005, x * m2 + 0.0
05);

                /* do a little consistancy checking */
        }

        /* if everything was ok, reassure the nervous captain */
        if (f)
                printf("All devices functional\n");
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)destruct.c       8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/destruct.c,v 1.4 1999/11/30 03:49:46 billf Exp $
 * $DragonFly: src/games/trek/destruct.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/*
**  Self Destruct Sequence
**
**      The computer starts up the self destruct sequence.  Obviously,
**      if the computer is out nothing can happen.  You get a countdown
**      and a request for password.  This must match the password that
**      you entered at the start of the game.
**
**      You get to destroy things when you blow up; hence, it is
**      possible to win the game by destructing if you take the last
**      Klingon with you.
**
**      By the way, the \032 in the message is a ^Z, which is because
**      the terminal in my office is an ADM-3, which uses that char-
**      acter to clear the screen.  I also stick in a \014 (form feed)
**      because that clears some other screens.
**
**      Uses trace flag 41
*/

void
destruct(__unused int unused)
```

```
{
        char            checkpass[15];
        int     i, j;
        double          zap;

        if (damaged(COMPUTER)) {
                out(COMPUTER);
                return;
        }
        printf("\n\07 --- WORKING ---\07\n");
        sleep(3);
        /* output the count 10 9 8 7 6 */
        for (i = 10; i > 5; i--)
        {
                for (j = 10;  j > i; j--)
                        printf(" ");
                printf("%d\n", i);
                sleep(1);
        }
        /* check for password on new line only */
        skiptonl(0);
        getstrpar("Enter password verification", checkpass, 14, 0);
        sleep(2);
        if (!sequal(checkpass, Game.passwd)) {
                printf("Self destruct sequence aborted\n");
                return;
        }
        printf("Password verified; self destruct sequence continues:\n");
        sleep(2);
        /* output count 5 4 3 2 1 0 */
        for (i = 5; i >= 0; i--)
        {
                sleep(1);
                for (j = 5; j > i; j--)
                        printf(" ");
                printf("%d\n", i);
        }
        sleep(2);
        printf("\032\014***** %s destroyed *****\n", Ship.shipname);
        Game.killed = 1;
        /* let's see what we can blow up!!!! */
        zap = 20.0 * Ship.energy;
        Game.deaths += Ship.crew;
        for (i = 0; i < Etc.nkling; )
        {
                if (Etc.klingon[i].power * Etc.klingon[i].dist <= zap)
                        killk(Etc.klingon[i].x, Etc.klingon[i].y);
                else
                        i++;
        }
        /* if we didn't kill the last Klingon (detected by killk), */
        /* then we lose.... */
        lose(L_DSTRCT);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)dock.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/dock.c,v 1.4 1999/11/30 03:49:46 billf Exp $
 * $DragonFly: src/games/trek/dock.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  DOCK TO STARBASE
**
**      The starship is docked to a starbase.  For this to work you
**      must be adjacent to a starbase.
**
**      You get your supplies replenished and your captives are
**      disembarked.  Note that your score is updated now, not when
**      you actually take the captives.
**
**      Any repairs that need to be done are rescheduled to take
**      place sooner.  This provides for the faster repairs when you
**      are docked.
*/

void
dock(__unused int unused)
{
        int             i, j;
        int                     ok;
        struct event    *e;

        if (Ship.cond == DOCKED)
```

```
        {
                printf("Chekov: But captain, we are already docked\n");
                return;
        }
        /* check for ok to dock, i.e., adjacent to a starbase */
        ok = 0;
        for (i = Ship.sectx − 1; i <= Ship.sectx + 1 && !ok; i++)
        {
                if (i < 0 || i >= NSECTS)
                        continue;
                for (j = Ship.secty − 1; j <= Ship.secty + 1; j++)
                {
                        if (j  < 0 || j >= NSECTS)
                                continue;
                        if (Sect[i][j] == BASE)
                        {
                                ok++;
                                break;
                        }
                }
        }
        if (!ok)
        {
                printf("Chekov: But captain, we are not adjacent to a starbase.\n");
                return;
        }

        /* restore resources */
        Ship.energy = Param.energy;
        Ship.torped = Param.torped;
        Ship.shield = Param.shield;
        Ship.crew = Param.crew;
        Game.captives += Param.brigfree − Ship.brigfree;
        Ship.brigfree = Param.brigfree;

        /* reset ship’s defenses */
        Ship.shldup = 0;
        Ship.cloaked = 0;
        Ship.cond = DOCKED;
        Ship.reserves = Param.reserves;

        /* recalibrate space inertial navigation system */
        Ship.sinsbad = 0;

        /* output any saved radio messages */
        dumpssradio();

        /* reschedule any device repairs */
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e−>evcode != E_FIXDV)
                        continue;
                reschedule(e, (e−>date − Now.date) * Param.dockfac);
        }
        return;
}


/*
**  LEAVE A STARBASE
**
```

```
**      This is the inverse of dock().  The main function it performs
**      is to reschedule any damages so that they will take longer.
*/

void
undock(__unused int unused)
{
        struct event    *e;
        int             i;

        if (Ship.cond != DOCKED)
        {
                printf("Sulu: Pardon me captain, but we are not docked.\n");
                return;
        }
        Ship.cond = GREEN;
        Move.free = 0;

        /* reschedule device repair times (again) */
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode != E_FIXDV)
                        continue;
                reschedule(e, (e->date - Now.date) / Param.dockfac);
        }
        return;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)dumpgame.c       8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/dumpgame.c,v 1.6 1999/11/30 03:49:46 billf Exp $
 * $DragonFly: src/games/trek/dumpgame.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

#include <fcntl.h>

# include       "trek.h"

/***  THIS CONSTANT MUST CHANGE AS THE DATA SPACES CHANGE ***/
# define        VERSION         2

struct dump
{
        char    *area;
        int     count;
};


struct dump     Dump_template[] =
{
        { (char *)&Ship,        sizeof (Ship)   },
        { (char *)&Now,         sizeof (Now)    },
        { (char *)&Param,       sizeof (Param)  },
        { (char *)&Etc,         sizeof (Etc)    },
        { (char *)&Game,        sizeof (Game)   },
        { (char *)Sect,         sizeof (Sect)   },
        { (char *)Quad,         sizeof (Quad)   },
        { (char *)&Move,        sizeof (Move)   },
        { (char *)Event,        sizeof (Event)  },
```

```
        { NULL,                 0               }
};

static bool     readdump(int);

/*
**  DUMP GAME
**
**      This routine dumps the game onto the file "trek.dump".  The
**      first two bytes of the file are a version number, which
**      reflects whether this image may be used.  Obviously, it must
**      change as the size, content, or order of the data structures
**      output change.
*/

void
dumpgame(__unused int unused)
{
        int                     version;
        int             fd;
        struct dump     *d;
        int             i;

        if ((fd = creat("trek.dump", 0644)) < 0) {
                printf("cannot dump\n");
                return;
        }
        version = VERSION;
        write(fd, &version, sizeof version);

        /* output the main data areas */
        for (d = Dump_template; d->area; d++)
        {
                write(fd, &d->area, sizeof d->area);
                i = d->count;
                write(fd, d->area, i);
        }

        close(fd);
}


/*
**  RESTORE GAME
**
**      The game is restored from the file "trek.dump".  In order for
**      this to succeed, the file must exist and be readable, must
**      have the correct version number, and must have all the appro-
**      priate data areas.
**
**      Return value is zero for success, one for failure.
*/

bool
restartgame(void)
{
        int     fd;
        int             version;

        if ((fd = open("trek.dump", O_RDONLY)) < 0 ||
            read(fd, &version, sizeof version) != sizeof version ||
            version != VERSION ||
            readdump(fd))
        {
                printf("cannot restart\n");
                close(fd);
                return (1);
        }

        close(fd);
        return (0);
}


/*
**  READ DUMP
**
**      This is the business end of restartgame().  It reads in the
**      areas.
**
**      Returns zero for success, one for failure.
*/

static bool
readdump(int fd1)
{
        int             fd;
        struct dump     *d;
        int             i;
        long                    junk;

        fd = fd1;

        for (d = Dump_template; d->area; d++)
        {
                if (read(fd, &junk, sizeof junk) != (sizeof junk))
                        return (1);
                if ((char *)junk != d->area)
                        return (1);
                i = d->count;
                if (read(fd, d->area, i) != i)
                        return (1);
        }

        /* make quite certain we are at EOF */
        return (read(fd, &junk, 1));
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)dumpme.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/dumpme.c,v 1.4 1999/11/30 03:49:47 billf Exp $
 * $DragonFly: src/games/trek/dumpme.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Dump the starship somewhere in the galaxy
**
**      Parameter is zero if bounce off of negative energy barrier,
**      one if through a black hole
**
**      Note that the quadrant is NOT initialized here.  This must
**      be done from the calling routine.
**
**      Repair of devices must be deferred.
*/

void
dumpme(int flag)
{
        int             f;
        double                  x = 0;
        struct event    *e;
        int             i;

        f = flag;
        Ship.quadx = ranf(NQUADS);
        Ship.quady = ranf(NQUADS);
```

```
        Ship.sectx = ranf(NSECTS);
        Ship.secty = ranf(NSECTS);
        x += 1.5 * franf();
        Move.time += x;
        if (f)
        {
                printf("%s falls into a black hole.\n", Ship.shipname);
        }
        else
        {
                printf("Computer applies full reverse power to avoid hitting the\n");
                printf("  negative energy barrier.  A space warp was entered.\n");
        }
        /* bump repair dates forward */
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode != E_FIXDV)
                        continue;
                reschedule(e, (e->date - Now.date) + x);
        }
        events(1);
        printf("You are now in quadrant %d,%d.  It is stardate %.2f\n",
                Ship.quadx, Ship.quady, Now.date);
        Move.time = 0;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)dumpssradio.c    8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/dumpssradio.c,v 1.4 1999/11/30 03:49:47 billf Exp $
 * $DragonFly: src/games/trek/dumpssradio.c,v 1.3 2006/09/07 21:19:44 pavalos Ex
p $
 */

# include       "trek.h"

/**
 **     output hidden distress calls
 **/

int
dumpssradio(void)
{
        struct event    *e;
        int             j;
        int             chkrest;

        chkrest = 0;
        for (j = 0; j < MAXEVENTS; j++)
        {
                e = &Event[j];
                /* if it is not hidden, then just ignore it */
                if ((e->evcode & E_HIDDEN) == 0)
                        continue;
                if (e->evcode & E_GHOST)
                {
                        unschedule(e);
                        printf("Starsystem %s in quadrant %d,%d is no longer distressed\n",
```

```
                                systemname(&Quad[e->x][e->y]), e->x, e->y);
                        continue;
                }

                switch (e->evcode)
                {

                  case E_KDESB:
                        printf("Starbase in quadrant %d,%d is under attack\n",
                                e->x, e->y);
                        chkrest++;
                        break;

                  case E_ENSLV:
                  case E_REPRO:
                        printf("Starsystem %s in quadrant %d,%d is distressed\n",
                                systemname(&Quad[e->x][e->y]), e->x, e->y);
                        chkrest++;
                        break;

                }
        }

        return (chkrest);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)events.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/events.c,v 1.4 1999/11/30 03:49:47 billf Exp $
 * $DragonFly: src/games/trek/events.c,v 1.4 2008/04/20 13:44:24 swildner Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/*
**  CAUSE TIME TO ELAPSE
**
**      This routine does a hell of a lot.  It elapses time, eats up
**      energy, regenerates energy, processes any events that occur,
**      and so on.
*/

void
events(int t_warp)
/* t_warp:  set if called in a time warp */
{
        int             i;
        int                     j = 0;
        struct kling            *k;
        double                  rtime;
        double                  xdate;
        double                  idate;
        struct event            *ev = NULL;
        char                    *s;
        int                     ix, iy;
        struct quad     *q;
```

```
        struct event    *e;
        int                     evnum;
        int                     restcancel;

        /* if nothing happened, just allow for any Klingons killed */
        if (Move.time <= 0.0)
        {
                Now.time = Now.resource / Now.klings;
                return;
        }

        /* indicate that the cloaking device is now working */
        Ship.cloakgood = 1;

        /* idate is the initial date */
        idate = Now.date;

        /* schedule attacks if resting too long */
        if (Move.time > 0.5 && Move.resting)
                schedule(E_ATTACK, 0.5, 0, 0, 0);

        /* scan the event list */
        while (1)
        {
                restcancel = 0;
                evnum = -1;
                /* xdate is the date of the current event */
                xdate = idate + Move.time;

                /* find the first event that has happened */
                for (i = 0; i < MAXEVENTS; i++)
                {
                        e = &Event[i];
                        if (e->evcode == 0 || (e->evcode & E_GHOST))
                                continue;
                        if (e->date < xdate)
                        {
                                xdate = e->date;
                                ev = e;
                                evnum = i;
                        }
                }
                e = ev;

                /* find the time between events */
                rtime = xdate - Now.date;

                /* decrement the magic "Federation Resources" pseudo-variable */
                Now.resource -= Now.klings * rtime;
                /* and recompute the time left */
                Now.time = Now.resource / Now.klings;

                /* move us up to the next date */
                Now.date = xdate;

                /* check for out of time */
                if (Now.time <= 0.0)
                        lose(L_NOTIME);
#               ifdef xTRACE
                if (evnum >= 0 && Trace)
                        printf("xdate = %.2f, evcode %d params %d %d %d\n",
                                xdate, e->evcode, e->x, e->y, e->systemname);
```

```
#               endif

                /* if evnum < 0, no events occurred  */
                if (evnum < 0)
                        break;

                /* otherwise one did.  Find out what it is */
                switch (e->evcode & E_EVENT)
                {

                  case E_SNOVA:                 /* supernova */
                        /* cause the supernova to happen */
                        snova(-1, 0);
                        /* and schedule the next one */
                        xresched(e, E_SNOVA, 1);
                        break;

                  case E_LRTB:                  /* long range tractor beam */
                        /* schedule the next one */
                        xresched(e, E_LRTB, Now.klings);
                        /* LRTB cannot occur if we are docked */
                        if (Ship.cond != DOCKED)
                        {
                                /* pick a new quadrant */
                                i = ranf(Now.klings) + 1;
                                for (ix = 0; ix < NQUADS; ix++)
                                {
                                        for (iy = 0; iy < NQUADS; iy++)
                                        {
                                                q = &Quad[ix][iy];
                                                if (q->stars >= 0)
                                                        if ((i -= q->klings) <=
0)
                                                                break;
                                        }
                                        if (i <= 0)
                                                break;
                                }

                                /* test for LRTB to same quadrant */
                                if (Ship.quadx == ix && Ship.quady == iy)
                                        break;

                                /* nope, dump him in the new quadrant */
                                Ship.quadx = ix;
                                Ship.quady = iy;
                                printf("\n%s caught in long range tractor beam\n", Ship.ship
name);
                                printf("*** Pulled to quadrant %d,%d\n", Ship.quadx, Shi
p.quady);
                                Ship.sectx = ranf(NSECTS);
                                Ship.secty = ranf(NSECTS);
                                initquad(0);
                                /* truncate the move time */
                                Move.time = xdate - idate;
                        }
                        break;

                  case E_KATSB:                 /* Klingon attacks starbase */
                        /* if out of bases, forget it */
                        if (Now.bases <= 0)
                        {
```

```
                                unschedule(e);
                                break;
                        }

                        /* check for starbase and Klingons in same quadrant */
                        for (i = 0; i < Now.bases; i++)
                        {
                                ix = Now.base[i].x;
                                iy = Now.base[i].y;
                                /* see if a Klingon exists in this quadrant */
                                q = &Quad[ix][iy];
                                if (q->klings <= 0)
                                        continue;

                                /* see if already distressed */
                                for (j = 0; j < MAXEVENTS; j++)
                                {
                                        e = &Event[j];
                                        if ((e->evcode & E_EVENT) != E_KDESB)
                                                continue;
                                        if (e->x == ix && e->y == iy)
                                                break;
                                }
                                if (j < MAXEVENTS)
                                        continue;

                                /* got a potential attack */
                                break;
                        }
                        e = ev;
                        if (i >= Now.bases)
                        {
                                /* not now; wait a while and see if some Klingon
s move in */
                                reschedule(e, 0.5 + 3.0 * franf());
                                break;
                        }
                        /* schedule a new attack, and a destruction of the base
*/
                        xresched(e, E_KATSB, 1);
                        e = xsched(E_KDESB, 1, ix, iy, 0);

                        /* report it if we can */
                        if (!damaged(SSRADIO))
                        {
                                printf("\nUhura:  Captain, we have received a distress signal\n");
                                printf("  from the starbase in quadrant %d,%d.\n",
                                        ix, iy);
                                restcancel++;
                        }
                        else
                                /* SSRADIO out, make it so we can't see the dist
ress call */
                                /* but it's still there!!! */
                                e->evcode |= E_HIDDEN;
                        break;

                  case E_KDESB:                 /* Klingon destroys starbase */
                        unschedule(e);
                        q = &Quad[e->x][e->y];
                        /* if the base has mysteriously gone away, or if the Kli
ngon
```

```
                           got tired and went home, ignore this event */
                        if (q->bases <=0 || q->klings <= 0)
                                break;
                        /* are we in the same quadrant? */
                        if (e->x == Ship.quadx && e->y == Ship.quady)
                        {
                                /* yep, kill one in this quadrant */
                                printf("\nSpock: ");
                                killb(Ship.quadx, Ship.quady);
                        }
                        else
                                /* kill one in some other quadrant */
                                killb(e->x, e->y);
                        break;

                  case E_ISSUE:         /* issue a distress call */
                        xresched(e, E_ISSUE, 1);
                        /* if we already have too many, throw this one away */
                        if (Ship.distressed >= MAXDISTR)
                                break;
                        /* try a whole bunch of times to find something suitable
 */
                        for (i = 0; i < 100; i++)
                        {
                                ix = ranf(NQUADS);
                                iy = ranf(NQUADS);
                                q = &Quad[ix][iy];
                                /* need a quadrant which is not the current one,
                                   which has some stars which are inhabited and
                                   not already under attack, which is not
                                   supernova'ed, and which has some Klingons in
it */
                                if (!((ix == Ship.quadx && iy == Ship.quady) ||
q->stars < 0 ||
                                    (q->qsystemname & Q_DISTRESSED) ||
                                    (q->qsystemname & Q_SYSTEM) == 0 || q->kling
s <= 0))
                                        break;
                        }
                        if (i >= 100)
                                /* can't seem to find one; ignore this call */
                                break;

                        /* got one!!  Schedule its enslavement */
                        Ship.distressed++;
                        e = xsched(E_ENSLV, 1, ix, iy, q->qsystemname);
                        q->qsystemname = (e - Event) | Q_DISTRESSED;

                        /* tell the captain about it if we can */
                        if (!damaged(SSRADIO))
                        {
                                printf("\nUhura: Captain, starsystem %s in quadrant %d,%d is und
er attack\n",
                                        Systemname[e->systemname], ix, iy);
                                restcancel++;
                        }
                        else
                                /* if we can't tell him, make it invisible */
                                e->evcode |= E_HIDDEN;
                        break;

                  case E_ENSLV:         /* starsystem is enslaved */
```

```
                        unschedule(e);
                        /* see if current distress call still active */
                        q = &Quad[e->x][e->y];
                        if (q->klings <= 0)
                        {
                                /* no Klingons, clean up */
                                /* restore the system name */
                                q->qsystemname = e->systemname;
                                break;
                        }

                        /* play stork and schedule the first baby */
                        e = schedule(E_REPRO, Param.eventdly[E_REPRO] * franf(),
 e->x, e->y, e->systemname);

                        /* report the disaster if we can */
                        if (!damaged(SSRADIO))
                        {
                                printf("\nUhura:  We've lost contact with starsystem %s\n",
                                        Systemname[e->systemname]);
                                printf(" in quadrant %d,%d.\n",
                                        e->x, e->y);
                        }
                        else
                                e->evcode |= E_HIDDEN;
                        break;

                  case E_REPRO:         /* Klingon reproduces */
                        /* see if distress call is still active */
                        q = &Quad[e->x][e->y];
                        if (q->klings <= 0)
                        {
                                unschedule(e);
                                q->qsystemname = e->systemname;
                                break;
                        }
                        xresched(e, E_REPRO, 1);
                        /* reproduce one Klingon */
                        ix = e->x;
                        iy = e->y;
                        if (Now.klings == 127)
                                break;          /* full right now */
                        if (q->klings >= MAXKLQUAD)
                        {
                                /* this quadrant not ok, pick an adjacent one */
                                for (i = ix - 1; i <= ix + 1; i++)
                                {
                                        if (i < 0 || i >= NQUADS)
                                                continue;
                                        for (j = iy - 1; j <= iy + 1; j++)
                                        {
                                                if (j < 0 || j >= NQUADS)
                                                        continue;
                                                q = &Quad[i][j];
                                                /* check for this quad ok (not f
ull & no snova) */
                                                if (q->klings >= MAXKLQUAD || q-
>stars < 0)
                                                        continue;
                                                break;
                                        }
                                        if (j <= iy + 1)
                                        break;
                                }
                                if (j > iy + 1)
                                        /* cannot create another yet */
                                        break;
                                ix = i;
                                iy = j;
                        }
                        /* deliver the child */
                        q->klings++;
                        Now.klings++;
                        if (ix == Ship.quadx && iy == Ship.quady)
                        {
                                /* we must position Klingon */
                                sector(&ix, &iy);
                                Sect[ix][iy] = KLINGON;
                                k = &Etc.klingon[Etc.nkling++];
                                k->x = ix;
                                k->y = iy;
                                k->power = Param.klingpwr;
                                k->srndreq = 0;
                                compkldist(Etc.klingon[0].dist == Etc.klingon[0]
.avgdist ? 0 : 1);
                        }

                        /* recompute time left */
                        Now.time = Now.resource / Now.klings;
                        break;

                  case E_SNAP:          /* take a snapshot of the galaxy */
                        xresched(e, E_SNAP, 1);
                        s = Etc.snapshot;
                        s = bmove(Quad, s, sizeof (Quad));
                        s = bmove(Event, s, sizeof (Event));
                        s = bmove(&Now, s, sizeof (Now));
                        Game.snap = 1;
                        break;

                  case E_ATTACK:        /* Klingons attack during rest period */
                        if (!Move.resting)
                        {
                                unschedule(e);
                                break;
                        }
                        attack(1);
                        reschedule(e, 0.5);
                        break;

                  case E_FIXDV:
                        i = e->systemname;
                        unschedule(e);

                        /* de-damage the device */
                        printf("%s reports repair work on the %s finished.\n",
                                Device[i].person, Device[i].name);

                        /* handle special processing upon fix */
                        switch (i)
                        {

                          case LIFESUP:
                                Ship.reserves = Param.reserves;
```

```
                                break;

                          case SINS:
                                if (Ship.cond == DOCKED)
                                        break;
                                printf("Spock has tried to recalibrate your Space Internal Navigatio
n System,\n");
                                printf("  but he has no standard base to calibrate to.  Suggest you g
et\n");
                                printf("  to a starbase immediately so that you can properly recalibr
ate.\n");
                                Ship.sinsbad = 1;
                                break;

                          case SSRADIO:
                                restcancel = dumpssradio();
                                break;
                        }
                        break;

                  default:
                        break;
                }

                if (restcancel && Move.resting && getynpar("Spock: Shall we cancel our re
st period"))
                        Move.time = xdate - idate;

        }

        /* unschedule an attack during a rest period */
        if ((e = Now.eventptr[E_ATTACK]))
                unschedule(e);

        if (!t_warp)
        {
                /* eat up energy if cloaked */
                if (Ship.cloaked)
                        Ship.energy -= Param.cloakenergy * Move.time;

                /* regenerate resources */
                rtime = 1.0 - exp(-Param.regenfac * Move.time);
                Ship.shield += (Param.shield - Ship.shield) * rtime;
                Ship.energy += (Param.energy - Ship.energy) * rtime;

                /* decrement life support reserves */
                if (damaged(LIFESUP) && Ship.cond != DOCKED)
                        Ship.reserves -= Move.time;
        }
        return;
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)externs.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/externs.c,v 1.2 1999/11/30 03:49:47 billf Exp $
 * $DragonFly: src/games/trek/externs.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**      global variable definitions
*/

struct device   Device[NDEV] =
{
        { "warp drive",         "Scotty" },
        { "S.R. scanners",      "Scotty" },
        { "L.R. scanners",      "Scotty" },
        { "phasers",            "Sulu" },
        { "photon tubes",       "Sulu" },
        { "impulse engines",    "Scotty" },
        { "shield control",     "Sulu" },
        { "computer",           "Spock" },
        { "subspace radio",     "Uhura" },
        { "life support",       "Scotty" },
        { "navigation system",  "Chekov" },
        { "cloaking device",    "Scotty" },
        { "transporter",        "Scotty" },
        { "shuttlecraft",       "Scotty" },
        { "*ERR 14*",           "Nobody" },
        { "*ERR 15*",           "Nobody" }
};
```

```c
const char      *Systemname[NINHAB] =
{
        "ERROR",
        "Talos IV",
        "Rigel III",
        "Deneb VII",
        "Canopus V",
        "Icarus I",
        "Prometheus II",
        "Omega VII",
        "Elysium I",
        "Scalos IV",
        "Procyon IV",
        "Arachnid I",
        "Argo VIII",
        "Triad III",
        "Echo IV",
        "Nimrod III",
        "Nemisis IV",
        "Centarurus I",
        "Kronos III",
        "Spectros V",
        "Beta III",
        "Gamma Tranguli VI",
        "Pyris III",
        "Triachus",
        "Marcus XII",
        "Kaland",
        "Ardana",
        "Stratos",
        "Eden",
        "Arrikis",
        "Epsilon Eridani IV",
        "Exo III"
};
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)getcodi.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/getcodi.c,v 1.2 1999/11/30 03:49:48 billf Exp $
 * $DragonFly: src/games/trek/getcodi.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/*
**  get course and distance
**
**      The user is asked for a course and distance.  This is used by
**      move, impulse, and some of the computer functions.
**
**      The return value is zero for success, one for an invalid input
**      (meaning to drop the request).
*/

bool
getcodi(int *co, double *di)
{

        *co = getintpar("Course");

        /* course must be in the interval [0, 360] */
        if (*co < 0 || *co > 360)
                return (1);
        *di = getfltpar("Distance");

        /* distance must be in the interval [0, 15] */
```

```c
        if (*di <= 0.0 || *di > 15.0)
                return (1);

        /* good return */
        return (0);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)getpar.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/getpar.c,v 1.5 1999/11/30 03:49:48 billf Exp $
 * $DragonFly: src/games/trek/getpar.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

static bool     testterm(void);

/**
 **     get integer parameter
 **/

int
getintpar(const char *s)
{
        int     i;
        int             n;

        while (1)
        {
                if (testnl() && s)
                        printf("%s: ", s);
                i = scanf("%d", &n);
                if (i < 0)
                        exit(1);
                if (i > 0 && testterm())
                        return (n);
                printf("invalid input; please enter an integer\n");
```

```
                skiptonl(0);
        }
}

/**
 **     get floating parameter
 **/

double
getfltpar(const char *s)
{
        int             i;
        double                  d;

        while (1)
        {
                if (testnl() && s)
                        printf("%s: ", s);
                i = scanf("%lf", &d);
                if (i < 0)
                        exit(1);
                if (i > 0 && testterm())
                        return (d);
                printf("invalid input; please enter a double\n");
                skiptonl(0);
        }
}

/**
 **     get yes/no parameter
 **/

struct cvntab   Yntab[] =
{
        { "y",  "es",   (void (*)(int))1,       0 },
        { "n",  "o",    (void (*)(int))0,       0 },
        { NULL, NULL,   NULL,                   0 }
};

long
getynpar(const char *s)
{
        struct cvntab           *r;

        r = getcodpar(s, Yntab);
        return ((long) r->value);
}


/**
 **     get coded parameter
 **/

struct cvntab *
getcodpar(const char *s, struct cvntab tab[])
{
        char                            input[100];
        struct cvntab           *r;
        int                             flag;
        char                    *p;
        const char              *q;
        int                             c;
        int                             f;

        flag = 0;
        while (1)
        {
                flag |= (f = testnl());
                if (flag)
                        printf("%s: ", s);
                if (f)
                        cgetc(0);               /* throw out the newline */
                scanf("%*[ \t;]");
                if ((c = scanf("%[^ \t;\n]", input)) < 0)
                        exit(1);
                if (c == 0)
                        continue;
                flag = 1;

                /* if command list, print four per line */
                if (input[0] == '?' && input[1] == 0)
                {
                        c = 4;
                        for (r = tab; r->abrev; r++)
                        {
                                strcpy(input, r->abrev);
                                strcat(input, r->full);
                                printf("%14.14s", input);
                                if (--c > 0)
                                        continue;
                                c = 4;
                                printf("\n");
                        }
                        if (c != 4)
                                printf("\n");
                        continue;
                }

                /* search for in table */
                for (r = tab; r->abrev; r++)
                {
                        p = input;
                        for (q = r->abrev; *q; q++)
                                if (*p++ != *q)
                                        break;
                        if (!*q)
                        {
                                for (q = r->full; *p && *q; q++, p++)
                                        if (*p != *q)
                                                break;
                                if (!*p || !*q)
                                        break;
                        }
                }

                /* check for not found */
                if (!r->abrev)
                {
                        printf("invalid input; ? for valid inputs\n");
                        skiptonl(0);
                }
                else
                        return (r);
        }
```

```
}


/**
 **     get string parameter
 **/

void
getstrpar(const char *s, char *r, int l, const char *t)
{
        int     i;
        char            format[20];
        int     f;

        if (t == 0)
                t = " \t\n;";
        sprintf(format, "%%%d[^%s]", l, t);
        while (1)
        {
                if ((f = testnl()) && s)
                        printf("%s: ", s);
                if (f)
                        cgetc(0);
                scanf("%*[\t ;]");
                i = scanf(format, r);
                if (i < 0)
                        exit(1);
                if (i != 0)
                        return;
        }
}


/**
 **     test if newline is next valid character
 **/

bool
testnl(void)
{
        char            c;

        while ((c = cgetc(0)) != '\n')
                if ((c >= '0' && c <= '9') || c == '.' || c == '!' ||
                                (c >= 'A' && c <= 'Z') ||
                                (c >= 'a' && c <= 'z') || c == '-')
                {
                        ungetc(c, stdin);
                        return(0);
                }
        ungetc(c, stdin);
        return (1);
}


/**
 **     scan for newline
 **/

void
skiptonl(char c)
{
        while (c != '\n')
                if (!(c = cgetc(0)))
                        return;
        ungetc('\n', stdin);
        return;
}


/**
 **     test for valid terminator
 **/

static bool
testterm(void)
{
        char            c;

        if (!(c = cgetc(0)))
                return (1);
        if (c == '.')
                return (0);
        if (c == '\n' || c == ';')
                ungetc(c, stdin);
        return (1);
}


/*
**  TEST FOR SPECIFIED DELIMETER
**
**      The standard input is scanned for the parameter.  If found,
**      it is thrown away and non-zero is returned.  If not found,
**      zero is returned.
*/

bool
readdelim(char d)
{
        char    c;

        while ((c = cgetc(0)))
        {
                if (c == d)
                        return (1);
                if (c == ' ')
                        continue;
                ungetc(c, stdin);
                break;
        }
        return (0);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)help.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/help.c,v 1.4 1999/11/30 03:49:48 billf Exp $
 * $DragonFly: src/games/trek/help.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  call starbase for help
**
**      First, the closest starbase is selected.  If there is a
**      a starbase in your own quadrant, you are in good shape.
**      This distance takes quadrant distances into account only.
**
**      A magic number is computed based on the distance which acts
**      as the probability that you will be rematerialized.  You
**      get three tries.
**
**      When it is determined that you should be able to be remater-
**      ialized (i.e., when the probability thing mentioned above
**      comes up positive), you are put into that quadrant (anywhere).
**      Then, we try to see if there is a spot adjacent to the star-
**      base.  If not, you can't be rematerialized!!!  Otherwise,
**      it drops you there.  It only tries five times to find a spot
**      to drop you.  After that, it's your problem.
*/

const char      *Cntvect[3] =
{"first", "second", "third"};
```

```
void
help(__unused int unused)
{
        int             i;
        double                  dist, x;
        int             dx, dy;
        int                     j, l = 0;

        /* check to see if calling for help is reasonable ... */
        if (Ship.cond == DOCKED) {
                printf("Uhura: But Captain, we're already docked\n");
                return;
        }
        /* or possible */
        if (damaged(SSRADIO)) {
                out(SSRADIO);
                return;
        }
        if (Now.bases <= 0) {
                printf("Uhura: I'm not getting any response from starbase\n");
                return;
        }
        /* tut tut, there goes the score */
        Game.helps += 1;

        /* find the closest base */
        dist = 1e50;
        if (Quad[Ship.quadx][Ship.quady].bases <= 0)
        {
                /* there isn't one in this quadrant */
                for (i = 0; i < Now.bases; i++)
                {
                        /* compute distance */
                        dx = Now.base[i].x - Ship.quadx;
                        dy = Now.base[i].y - Ship.quady;
                        x = dx * dx + dy * dy;
                        x = sqrt(x);

                        /* see if better than what we already have */
                        if (x < dist)
                        {
                                dist = x;
                                l = i;
                        }
                }

                /* go to that quadrant */
                Ship.quadx = Now.base[l].x;
                Ship.quady = Now.base[l].y;
                initquad(1);
        }
        else
        {
                dist = 0.0;
        }

        /* dematerialize the Enterprise */
        Sect[Ship.sectx][Ship.secty] = EMPTY;
        printf("Starbase in %d,%d responds\n", Ship.quadx, Ship.quady);

        /* this next thing acts as a probability that it will work */
        x = pow(1.0 - pow(0.94, dist), 0.3333333);
        /* attempt to rematerialize */
        for (i = 0; i < 3; i++)
        {
                sleep(2);
                printf("%s attempt to rematerialize ", Cntvect[i]);
                if (franf() > x)
                {
                        /* ok, that's good.  let's see if we can set her down */
                        for (j = 0; j < 5; j++)
                        {
                                dx = Etc.starbase.x + ranf(3) − 1;
                                if (dx < 0 || dx >= NSECTS)
                                        continue;
                                dy = Etc.starbase.y + ranf(3) − 1;
                                if (dy < 0 || dy >= NSECTS || Sect[dx][dy] != EM
PTY)
                                        continue;
                                break;
                        }
                        if (j < 5)
                        {
                                /* found an empty spot */
                                printf("succeeds\n");
                                Ship.sectx = dx;
                                Ship.secty = dy;
                                Sect[dx][dy] = Ship.ship;
                                dock(0);
                                compkldist(0);
                                return;
                        }
                        /* the starbase must have been surrounded */
                }
                printf("fails\n");
        }

        /* one, two, three strikes, you're out */
        lose(L_NOHELP);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)impulse.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/impulse.c,v 1.4 1999/11/30 03:49:48 billf Exp $
 * $DragonFly: src/games/trek/impulse.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/**
 **     move under impulse power
 **/

void
impulse(__unused int unused)
{
        int                     course;
        int             power;
        double                  dist, p_time;
        int             percent;

        if (Ship.cond == DOCKED) {
                printf("Scotty: Sorry captain, but we are still docked.\n");
                return;
        }
        if (damaged(IMPULSE)) {
                out(IMPULSE);
                return;
        }
        if (getcodi(&course, &dist))
                return;
```

```c
        power = 20 + 100 * dist;
        percent = 100 * power / Ship.energy + 0.5;
        if (percent >= 85)
        {
                printf("Scotty: That would consume %d%% of our remaining energy.\n",
                        percent);
                if (!getynpar("Are you sure that is wise"))
                        return;
                printf("Aye aye, sir\n");
        }
        p_time = dist / 0.095;
        percent = 100 * p_time / Now.time + 0.5;
        if (percent >= 85)
        {
                printf("Spock: That would take %d%% of our remaining time.\n",
                        percent);
                if (!getynpar("Are you sure that is wise"))
                        return;
                printf("(He's finally gone mad)\n");
        }
        Move.time = move(0, course, p_time, 0.095);
        Ship.energy -= 20 + 100 * Move.time * 0.095;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)initquad.c       8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/initquad.c,v 1.4 1999/11/30 03:49:49 billf Exp $
 * $DragonFly: src/games/trek/initquad.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Paramize Quadrant Upon Entering
**
**      A quadrant is initialized from the information held in the
**      Quad matrix.  Basically, everything is just initialized
**      randomly, except for the starship, which goes into a fixed
**      sector.
**
**      If there are Klingons in the quadrant, the captain is informed
**      that the condition is RED, and he is given a chance to put
**      his shields up if the computer is working.
**
**      The flag `f' is set to disable the check for condition red.
**      This mode is used in situations where you know you are going
**      to be docked, i.e., abandon() and help().
*/

void
initquad(int f)
{
        int             i, j;
        int                     rx, ry;
        int                     nbases, nstars;
```

```
        struct quad     *q;
        int                     nholes;

        q = &Quad[Ship.quadx][Ship.quady];

        /* ignored supernova'ed quadrants (this is checked again later anyway */
        if (q->stars < 0)
                return;
        Etc.nkling = q->klings;
        nbases = q->bases;
        nstars = q->stars;
        nholes = q->holes;

        /* have we blundered into a battle zone w/ shields down? */
        if (Etc.nkling > 0 && !f)
        {
                printf("Condition RED\n");
                Ship.cond = RED;
                if (!damaged(COMPUTER))
                        shield(1);
        }

        /* clear out the quadrant */
        for (i = 0; i < NSECTS; i++)
                for (j = 0; j < NSECTS; j++)
                        Sect[i][j] = EMPTY;

        /* initialize Enterprise */
        Sect[Ship.sectx][Ship.secty] = Ship.ship;

        /* initialize Klingons */
        for (i = 0; i < Etc.nkling; i++)
        {
                sector(&rx, &ry);
                Sect[rx][ry] = KLINGON;
                Etc.klingon[i].x = rx;
                Etc.klingon[i].y = ry;
                Etc.klingon[i].power = Param.klingpwr;
                Etc.klingon[i].srndreq = 0;
        }
        compkldist(1);

        /* initialize star base */
        if (nbases > 0)
        {
                sector(&rx, &ry);
                Sect[rx][ry] = BASE;
                Etc.starbase.x = rx;
                Etc.starbase.y = ry;
        }

        /* initialize inhabited starsystem */
        if (q->qsystemname != 0)
        {
                sector(&rx, &ry);
                Sect[rx][ry] = INHABIT;
                nstars -= 1;
        }

        /* initialize black holes */
        for (i = 0; i < nholes; i++)
        {
                sector(&rx, &ry);
                Sect[rx][ry] = HOLE;
        }

        /* initialize stars */
        for (i = 0; i < nstars; i++)
        {
                sector(&rx, &ry);
                Sect[rx][ry] = STAR;
        }
        Move.newquad = 1;
}

void
sector(int *x, int *y)
{
        int             i, j;

        do
        {
                i = ranf(NSECTS);
                j = ranf(NSECTS);
        } while (Sect[i][j] != EMPTY);
        *x = i;
        *y = j;
        return;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)kill.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/kill.c,v 1.4 1999/11/30 03:49:49 billf Exp $
 * $DragonFly: src/games/trek/kill.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  KILL KILL KILL !!!
**
**      This file handles the killing off of almost anything.
*/

/*
**  Handle a Klingon's death
**
**      The Klingon at the sector given by the parameters is killed
**      and removed from the Klingon list.  Notice that it is not
**      removed from the event list; this is done later, when the
**      the event is to be caught.  Also, the time left is recomputed,
**      and the game is won if that was the last klingon.
*/

void
killk(int ix, int iy)
{
        int             i;

        printf(" *** Klingon at %d,%d destroyed ***\n", ix, iy);
```

```
        /* remove the scoundrel */
        Now.klings -= 1;
        Sect[ix][iy] = EMPTY;
        Quad[Ship.quadx][Ship.quady].klings -= 1;
        /* %%% IS THIS SAFE???? %%% */
        Quad[Ship.quadx][Ship.quady].scanned -= 100;
        Game.killk += 1;

        /* find the Klingon in the Klingon list */
        for (i = 0; i < Etc.nkling; i++)
                if (ix == Etc.klingon[i].x && iy == Etc.klingon[i].y)
                {
                        /* purge him from the list */
                        Etc.nkling -= 1;
                        for (; i < Etc.nkling; i++)
                                bmove(&Etc.klingon[i+1], &Etc.klingon[i], sizeof
 Etc.klingon[i]);
                        break;
                }

        /* find out if that was the last one */
        if (Now.klings <= 0)
                win();

        /* recompute time left */
        Now.time = Now.resource / Now.klings;
        return;
}


/*
**  handle a starbase's death
*/

void
killb(int qx, int qy)
{
        struct quad     *q;
        struct xy       *b;

        q = &Quad[qx][qy];

        if (q->bases <= 0)
                return;
        if (!damaged(SSRADIO))
        {
                /* then update starchart */
                if (q->scanned < 1000)
                        q->scanned -= 10;
                else
                        if (q->scanned > 1000)
                                q->scanned = -1;
        }
        q->bases = 0;
        Now.bases -= 1;
        for (b = Now.base; ; b++)
                if (qx == b->x && qy == b->y)
                        break;
        bmove(&Now.base[Now.bases], b, sizeof *b);
        if (qx == Ship.quadx && qy == Ship.quady)
        {
                Sect[Etc.starbase.x][Etc.starbase.y] = EMPTY;
                if (Ship.cond == DOCKED)
                        undock(0);
                printf("Starbase at %d,%d destroyed\n", Etc.starbase.x, Etc.starbase.y);
        }
        else
        {
                if (!damaged(SSRADIO))
                {
                        printf("Uhura: Starfleet command reports that the starbase in\n");
                        printf("   quadrant %d,%d has been destroyed\n", qx, qy);
                }
                else
                        schedule(E_KATSB | E_GHOST, 1e50, qx, qy, 0);
        }
}


/**
 **     kill an inhabited starsystem
 **/

void
kills(int x, int y, int f)
/* x,y:  quad coords if f == 0, else sector coords */
/* f != 0 -- this quad;  f < 0 -- Enterprise's fault */
{
        struct quad     *q;
        struct event    *e;
        const char      *name;

        if (f)
        {
                /* current quadrant */
                q = &Quad[Ship.quadx][Ship.quady];
                Sect[x][y] = EMPTY;
                name = systemname(q);
                if (name == 0)
                        return;
                printf("Inhabited starsystem %s at %d,%d destroyed\n",
                        name, x, y);
                if (f < 0)
                        Game.killinhab += 1;
        }
        else
        {
                /* different quadrant */
                q = &Quad[x][y];
        }
        if (q->qsystemname & Q_DISTRESSED)
        {
                /* distressed starsystem */
                e = &Event[q->qsystemname & Q_SYSTEM];
                printf("Distress call for %s invalidated\n",
                        Systemname[e->systemname]);
                unschedule(e);
        }
        q->qsystemname = 0;
        q->stars -= 1;
}


/**
```

```
 **     "kill" a distress call
 **/

void
killd(int x, int y, int f)
/* x,y:  quadrant coordinates */
/* f:  set if user is to be informed */
{
        struct event    *e;
        int             i;
        struct quad     *q;

        q = &Quad[x][y];
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->x != x || e->y != y)
                        continue;
                switch (e->evcode)
                {
                  case E_KDESB:
                        if (f)
                        {
                                printf("Distress call for starbase in %d,%d nullified\n",
                                        x, y);
                                unschedule(e);
                        }
                        break;

                  case E_ENSLV:
                  case E_REPRO:
                        if (f)
                        {
                                printf("Distress call for %s in quadrant %d,%d nullified\n",
                                        Systemname[e->systemname], x, y);
                                q->qsystemname = e->systemname;
                                unschedule(e);
                        }
                        else
                        {
                                e->evcode |= E_GHOST;
                        }
                }
        }
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)klmove.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/klmove.c,v 1.4 1999/11/30 03:49:49 billf Exp $
 * $DragonFly: src/games/trek/klmove.c,v 1.4 2006/10/08 17:11:30 pavalos Exp $
 */

# include       "trek.h"

/*
**  Move Klingons Around
**
**      This is a largely incomprehensible block of code that moves
**      Klingons around in a quadrant.  It was written in a very
**      "program as you go" fashion, and is a prime candidate for
**      rewriting.
**
**      The flag 'fl' is zero before an attack, one after an attack,
**      and two if you are leaving a quadrant.  This serves to
**      change the probability and distance that it moves.
**
**      Basically, what it will try to do is to move a certain number
**      of steps either toward you or away from you.  It will avoid
**      stars whenever possible.  Nextx and nexty are the next
**      sector to move to on a per-Klingon basis; they are roughly
**      equivalent to Ship.sectx and Ship.secty for the starship.  Lookx and
**      looky are the sector that you are going to look at to see
**      if you can move their.  Dx and dy are the increment.  Fudgex
**      and fudgey are the things you change around to change your
**      course around stars.
*/
```

```
void
klmove(int fl)
{
        int                     n;
        struct kling    *k;
        double                  dx, dy;
        int                     nextx, nexty;
        int             lookx, looky;
        int                     motion;
        int                     fudgex, fudgey;
        int                     qx, qy;
        double                  bigger;
        int                     i;

#       ifdef xTRACE
        if (Trace)
                printf("klmove: fl = %d, Etc.nkling = %d\n", fl, Etc.nkling);
#       endif
        for (n = 0; n < Etc.nkling; n++)
        {
                k = &Etc.klingon[n];
                i = 100;
                if (fl)
                        i = 100.0 * k->power / Param.klingpwr;
                if (ranf(i) >= Param.moveprob[2 * Move.newquad + fl])
                        continue;
                /* compute distance to move */
                motion = ranf(75) - 25;
                motion *= k->avgdist * Param.movefac[2 * Move.newquad + fl];
                /* compute direction */
                dx = Ship.sectx - k->x + ranf(3) - 1;
                dy = Ship.secty - k->y + ranf(3) - 1;
                bigger = dx;
                if (dy > bigger)
                        bigger = dy;
                if (bigger == 0.0)
                        bigger = 1.0;
                dx = dx / bigger + 0.5;
                dy = dy / bigger + 0.5;
                if (motion < 0)
                {
                        motion = -motion;
                        dx = -dx;
                        dy = -dy;
                }
                fudgex = fudgey = 1;
                /* try to move the klingon */
                nextx = k->x;
                nexty = k->y;
                for (; motion > 0; motion--)
                {
                        lookx = nextx + dx;
                        looky = nexty + dy;
                        if (lookx < 0 || lookx >= NSECTS || looky < 0 || looky >
= NSECTS)
                        {
                                /* new quadrant */
                                qx = Ship.quadx;
                                qy = Ship.quady;
                                if (lookx < 0)
                                        qx -= 1;
                                else
```

```
                                        if (lookx >= NSECTS)
                                                qx += 1;
                                if (looky < 0)
                                        qy -= 1;
                                else
                                        if (looky >= NSECTS)
                                                qy += 1;
                                if (qx < 0 || qx >= NQUADS || qy < 0 || qy >= NQ
UADS ||
                                                Quad[qx][qy].stars < 0 || Quad[q
x][qy].klings > MAXKLQUAD - 1)
                                        break;
                                if (!damaged(SRSCAN))
                                {
                                        printf("Klingon at %d,%d escapes to quadrant %d,%d\
n",
                                                k->x, k->y, qx, qy);
                                        motion = Quad[qx][qy].scanned;
                                        if (motion >= 0 && motion < 1000)
                                                Quad[qx][qy].scanned += 100;
                                        motion = Quad[Ship.quadx][Ship.quady].sc
anned;
                                        if (motion >= 0 && motion < 1000)
                                                Quad[Ship.quadx][Ship.quady].sca
nned -= 100;
                                }
                                Sect[k->x][k->y] = EMPTY;
                                Quad[qx][qy].klings += 1;
                                Etc.nkling -= 1;
                                bmove(&Etc.klingon[Etc.nkling], k, sizeof *k);
                                Quad[Ship.quadx][Ship.quady].klings -= 1;
                                k = 0;
                                break;
                        }
                        if (Sect[lookx][looky] != EMPTY)
                        {
                                lookx = nextx + fudgex;
                                if (lookx < 0 || lookx >= NSECTS)
                                        lookx = nextx + dx;
                                if (Sect[lookx][looky] != EMPTY)
                                {
                                        fudgex = -fudgex;
                                        looky = nexty + fudgey;
                                        if (looky < 0 || looky >= NSECTS || Sect
[lookx][looky] != EMPTY)
                                        {
                                                fudgey = -fudgey;
                                                break;
                                        }
                                }
                        }
                        nextx = lookx;
                        nexty = looky;
                }
                if (k && (k->x != nextx || k->y != nexty))
                {
                        if (!damaged(SRSCAN))
                                printf("Klingon at %d,%d moves to %d,%d\n",
                                        k->x, k->y, nextx, nexty);
                        Sect[k->x][k->y] = EMPTY;
                        Sect[k->x = nextx][k->y = nexty] = KLINGON;
                }
```

```
        }
        compkldist(0);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)lose.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/lose.c,v 1.2 1999/11/30 03:49:49 billf Exp $
 * $DragonFly: src/games/trek/lose.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/*
**  PRINT OUT LOSER MESSAGES
**
**      The messages are printed out, the score is computed and
**      printed, and the game is restarted.  Oh yeh, any special
**      actions which need be taken are taken.
*/

const char      *Losemsg[] =
{
        "You ran out of time",
        "You ran out of energy",
        "You have been destroyed",
        "You ran into the negative energy barrier",
        "You destroyed yourself by nova'ing that star",
        "You have been caught in a supernova",
        "You just suffocated in outer space",
        "You could not be rematerialized",
        "\n\032\014 ***\07 Ship's hull has imploded\07 ***",
        "You have burned up in a star",
        "Well, you destroyed yourself, but it didn't do any good",
        "You have been captured by Klingons and mercilessly tortured",
```

```
        "Your last crew member died",
};

void
lose(int why)
{
        Game.killed = 1;
        sleep(1);
        printf("\n%s\n", Losemsg[why - 1]);
        switch (why)
        {

          case L_NOTIME:
                Game.killed = 0;
                break;
        }
        Move.endgame = -1;
        score();
        skiptonl(0);
        longjmp(env, 1);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)lrscan.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/lrscan.c,v 1.4 1999/11/30 03:49:50 billf Exp $
 * $DragonFly: src/games/trek/lrscan.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  LONG RANGE OF SCANNERS
**
**      A summary of the quadrants that surround you is printed.  The
**      hundreds digit is the number of Klingons in the quadrant,
**      the tens digit is the number of starbases, and the units digit
**      is the number of stars.  If the printout is "///" it means
**      that that quadrant is rendered uninhabitable by a supernova.
**      It also updates the "scanned" field of the quadrants it scans,
**      for future use by the "chart" option of the computer.
*/

void
lrscan(__unused int unused)
{
        int                     i, j;
        struct quad             *q;

        if (check_out(LRSCAN))
        {
                return;
        }
        printf("Long range scan for quadrant %d,%d\n\n", Ship.quadx, Ship.quady);
```

```
        /* print the header on top */
        for (j = Ship.quady - 1; j <= Ship.quady + 1; j++)
        {
                if (j < 0 || j >= NQUADS)
                        printf("    ");
                else
                        printf("   %1d", j);
        }

        /* scan the quadrants */
        for (i = Ship.quadx - 1; i <= Ship.quadx + 1; i++)
        {
                printf("\n  -------------------\n");
                if (i < 0 || i >= NQUADS)
                {
                        /* negative energy barrier */
                        printf(" ! * ! * ! * !");
                        continue;
                }

                /* print the left hand margin */
                printf("%1d !", i);
                for (j = Ship.quady - 1; j <= Ship.quady + 1; j++)
                {
                        if (j < 0 || j >= NQUADS)
                        {
                                /* negative energy barrier again */
                                printf(" * !");
                                continue;
                        }
                        q = &Quad[i][j];
                        if (q->stars < 0)
                        {
                                /* supernova */
                                printf(" /// !");
                                q->scanned = 1000;
                                continue;
                        }
                        q->scanned = q->klings * 100 + q->bases * 10 + q->stars;
                        printf(" %3d !", q->scanned);
                }
        }
        printf("\n  -------------------\n");
        return;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#) Copyright (c) 1980, 1993 The Regents of the University of California.  A
ll rights reserved.
 * @(#)main.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/main.c,v 1.7.2.1 2001/03/05 12:11:14 kris Exp $
 * $DragonFly: src/games/trek/main.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

# define        PRIO            00      /* default priority */

unsigned int    Mother  = 51 + (51 << 8);

/*
**       ####  #####    #    ####          #####  ####   #####  #   #
**      #        #     # #   #   #           #    #   #  #      #  #
**       ###     #    #####  ####            #    ####   ###    ###
**          #    #    #   #  #  #            #    #  #   #      #  #
**      ####     #    #   #  #   #           #    #   #  #####  #   #
**
**      C version by Eric P. Allman 5/76 (U.C. Berkeley) with help
**              from Jeff Poskanzer and Pete Rubinstein.
**
**      I also want to thank everyone here at Berkeley who
**      where crazy enough to play the undebugged game.  I want to
**      particularly thank Nick Whyte, who made considerable
**      suggestions regarding the content of the game.  Why, I’ll
**      never forget the time he suggested the name for the
**      "capture" command.
```

```
**
**      Please send comments, questions, and suggestions about this
**              game to:
**                      Eric P. Allman
**                      Project INGRES
**                      Electronics Research Laboratory
**                      Cory Hall
**                      University of California
**                      Berkeley, California  94720
**
**      If you make ANY changes in the game, I sure would like to
**      know about them.  It is sort of an ongoing project for me,
**      and I very much want to put in any bug fixes and improvements
**      that you might come up with.
**
**      FORTRASH version by Kay R. Fisher (DEC) "and countless others".
**      That was adapted from the "original BASIC program" (ha!) by
**              Mike Mayfield (Centerline Engineering).
**
**      Additional inspiration taken from FORTRAN version by
**              David Matuszek and Paul Reynolds which runs on the CDC
**              7600 at Lawrence Berkeley Lab, maintained there by
**              Andy Davidson.  This version is also available at LLL
**              and at LMSC.  In all fairness, this version was the
**              major inspiration for this version of the game (trans−
**              lation:  I ripped off a whole lot of code).
**
**      Minor other input from the "Battelle Version 7A" by Joe Miller
**              (Graphics Systems Group, Battelle−Columbus Labs) and
**              Ross Pavlac (Systems Programmer, Battelle Memorial
**              Institute).  That version was written in December ’74
**              and extensively modified June ’75.  It was adapted
**              from the FTN version by Ron Williams of CDC Sunnyvale,
**              which was adapted from the Basic version distributed
**              by DEC.  It also had "neat stuff swiped" from T. T.
**              Terry and Jim Korp (University of Texas), Hicks (Penn
**              U.), and Rick Maus (Georgia Tech).  Unfortunately, it
**              was not as readable as it could have been and so the
**              translation effort was severely hampered.  None the
**              less, I got the idea of inhabited starsystems from this
**              version.
**
**      Permission is given for use, copying, and modification of
**              all or part of this program and related documentation,
**              provided that all reference to the authors are maintained.
**
**
**********************************************************************
**
**  NOTES TO THE MAINTAINER:
**
**      There is a compilation option xTRACE which must be set for any
**      trace information to be generated.  It is probably defined in
**      the version that you get.  It can be removed, however, if you
**      have trouble finding room in core.
**
**      Many things in trek are not as clear as they might be, but are
**      done to reduce space.  I compile with the −f and −O flags.  I
**      am constrained to running with non−separated I/D space, since
**      we don’t have doubleing point hardware here; even if we did, I
**      would like trek to be available to the large number of people
**      who either have an 11/40 or do not have FP hardware.  I also
**      found it desirable to make the code run reentrant, so this
**      added even more space constraints.
**
**      I use the portable C library to do my I/O.  This is done be-
**      cause I wanted the game easily transportable to other C
**      implementations, and because I was too lazy to do the doubleing
**      point input myself.  Little did I know.  The portable C library
**      released by Bell Labs has more bugs than you would believe, so
**      I ended up rewriting the whole blessed thing.  Trek excercises
**      many of the bugs in it, as well as bugs in some of the section
**      III UNIX routines.  We have fixed them here.  One main problem
**      was a bug in alloc() that caused it to always ask for a large
**      hunk of memory, which worked fine unless you were almost out,
**      which I inevitably was.  If you want the code for all of this
**      stuff, it is also available through me.
**
***********************************************************************
*/

jmp_buf env;

int
main(int argc, char **argv)
{
        /* extern FILE          *f_log; */
        char            opencode;
        int                     prio;
        int             ac;
        char            **av;

        /* revoke */
        setgid(getgid());

        av = argv;
        ac = argc;
        av++;
        srandomdev();
        opencode = 'w';
        prio = PRIO;

        while (ac > 1 && av[0][0] == '-')
        {
                switch (av[0][1])
                {
                  case 'a':     /* append to log file */
                        opencode = 'a';
                        break;

#               ifdef xTRACE
                  case 't':     /* trace */
                        if (getuid() != Mother)
                                goto badflag;
                        Trace++;
                        break;
#               endif

                  case 'p':     /* set priority */
                        if (getuid() != Mother)
                                goto badflag;
                        prio = atoi(av[0] + 2);
                        break;
```

```
                  default:
                  badflag:
                        printf("Invalid option: %s\n", av[0]);

                }
                ac--;
                av++;
        }
        if (ac > 2)
                syserr(0, "arg count");
                /*
        if (ac > 1)
                f_log = fopen(av[0], opencode);
                */

        printf("\n   * * *   S T A R   T R E K   * * *\n\nPress return to continue.\n");

        if (setjmp(env))
        {
                if ( !getynpar("Another game") )
                        exit(0);
        }
        do
        {
                setup();
                play();
        } while (getynpar("Another game"));

        fflush(stdout);
        return(0);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)move.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/move.c,v 1.6 1999/11/30 03:49:50 billf Exp $
 * $DragonFly: src/games/trek/move.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Move Under Warp or Impulse Power
**
**      `Ramflag' is set if we are to be allowed to ram stars,
**      Klingons, etc.  This is passed from warp(), which gets it from
**      either play() or ram().  Course is the course (0 -> 360) at
**      which we want to move.  `Speed' is the speed we
**      want to go, and `p_time' is the expected time.  It
**      can get cut short if a long range tractor beam is to occur.  We
**      cut short the move so that the user doesn't get docked time and
**      energy for distance which he didn't travel.
**
**      We check the course through the current quadrant to see that he
**      doesn't run into anything.  After that, though, space sort of
**      bends around him.  Note that this puts us in the awkward posi-
**      tion of being able to be dropped into a sector which is com-
**      pletely surrounded by stars.  Oh Well.
**
**      If the SINS (Space Inertial Navigation System) is out, we ran-
**      domize the course accordingly before ever starting to move.
**      We will still move in a straight line.
**
**      Note that if your computer is out, you ram things anyway.  In
```

```
**      other words, if your computer and sins are both out, you're in
**      potentially very bad shape.
**
**      Klingons get a chance to zap you as you leave the quadrant.
**      By the way, they also try to follow you (heh heh).
**
**      Return value is the actual amount of time used.
**
**
**      Uses trace flag 4.
*/

double
move(int ramflag, int course, double p_time, double speed)
{
        double                  angle;
        double                  x, y, dx, dy;
        int             ix, iy;
        double                  bigger;
        int                     n;
        int             i;
        double                  dist;
        double                  sectsize;
        double                  xn;
        double                  evtime;

        ix = iy = 0;
#       ifdef xTRACE
        if (Trace)
                printf("move: ramflag %d course %d time %.2f speed %.2f\n",
                        ramflag, course, p_time, speed);
#       endif
        sectsize = NSECTS;
        /* initialize delta factors for move */
        angle = course * 0.0174532925;
        if (damaged(SINS))
                angle += Param.navigcrud[1] * (franf() - 0.5);
        else
                if (Ship.sinsbad)
                        angle += Param.navigcrud[0] * (franf() - 0.5);
        dx = -cos(angle);
        dy = sin(angle);
        bigger = fabs(dx);
        dist = fabs(dy);
        if (dist > bigger)
                bigger = dist;
        dx /= bigger;
        dy /= bigger;

        /* check for long range tractor beams */
        /****  TEMPORARY CODE == DEBUGGING  ****/
        evtime = Now.eventptr[E_LRTB]->date - Now.date;
#       ifdef xTRACE
        if (Trace)
                printf("E.ep = %p, ->evcode = %d, ->date = %.2f, evtime = %.2f\n",
                        (void *)Now.eventptr[E_LRTB],
                        Now.eventptr[E_LRTB]->evcode,
                        Now.eventptr[E_LRTB]->date, evtime);
#       endif
        if (p_time > evtime && Etc.nkling < 3)
        {
                /* then we got a LRTB */
                evtime += 0.005;
                p_time = evtime;
        }
        else
                evtime = -1.0e50;
        dist = p_time * speed;

        /* move within quadrant */
        Sect[Ship.sectx][Ship.secty] = EMPTY;
        x = Ship.sectx + 0.5;
        y = Ship.secty + 0.5;
        xn = NSECTS * dist * bigger;
        n = xn + 0.5;
#       ifdef xTRACE
        if (Trace)
                printf("dx = %.2f, dy = %.2f, xn = %.2f, n = %d\n", dx, dy, xn, n);
#       endif
        Move.free = 0;

        for (i = 0; i < n; i++)
        {
                ix = (x += dx);
                iy = (y += dy);
#               ifdef xTRACE
                if (Trace)
                        printf("ix = %d, x = %.2f, iy = %d, y = %.2f\n", ix, x, iy, y);
#               endif
                if (x < 0.0 || y < 0.0 || x >= sectsize || y >= sectsize)
                {
                        /* enter new quadrant */
                        dx = Ship.quadx * NSECTS + Ship.sectx + dx * xn;
                        dy = Ship.quady * NSECTS + Ship.secty + dy * xn;
                        if (dx < 0.0)
                                ix = -1;
                        else
                                ix = dx + 0.5;
                        if (dy < 0.0)
                                iy = -1;
                        else
                                iy = dy + 0.5;
#                       ifdef xTRACE
                        if (Trace)
                                printf("New quad: ix = %d, iy = %d\n", ix, iy);
#                       endif
                        Ship.sectx = x;
                        Ship.secty = y;
                        compkldist(0);
                        Move.newquad = 2;
                        attack(0);
                        checkcond();
                        Ship.quadx = ix / NSECTS;
                        Ship.quady = iy / NSECTS;
                        Ship.sectx = ix % NSECTS;
                        Ship.secty = iy % NSECTS;
                        if (ix < 0 || Ship.quadx >= NQUADS || iy < 0 || Ship.qua
dy >= NQUADS)
                        {
                                if (!damaged(COMPUTER))
                                {
                                        dumpme(0);
                                }
                                else
```

```
                                        lose(L_NEGENB);
                                }
                                initquad(0);
                                n = 0;
                                break;
                        }
                        if (Sect[ix][iy] != EMPTY)
                        {
                                /* we just hit something */
                                if (!damaged(COMPUTER) && ramflag <= 0)
                                {
                                        ix = x - dx;
                                        iy = y - dy;
                                        printf("Computer reports navigation error; %s stopped at %d,%d\n
",
                                                Ship.shipname, ix, iy);
                                        Ship.energy -= Param.stopengy * speed;
                                        break;
                                }
                                /* test for a black hole */
                                if (Sect[ix][iy] == HOLE)
                                {
                                        /* get dumped elsewhere in the galaxy */
                                        dumpme(1);
                                        initquad(0);
                                        n = 0;
                                        break;
                                }
                                ram(ix, iy);
                                break;
                        }
                }
        if (n > 0)
        {
                dx = Ship.sectx - ix;
                dy = Ship.secty - iy;
                dist = sqrt(dx * dx + dy * dy) / NSECTS;
                p_time = dist / speed;
                if (evtime > p_time)
                        p_time = evtime;                /* spring the LRTB trap
*/
                Ship.sectx = ix;
                Ship.secty = iy;
        }
        Sect[Ship.sectx][Ship.secty] = Ship.ship;
        compkldist(0);
        return (p_time);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)nova.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/nova.c,v 1.4 1999/11/30 03:49:52 billf Exp $
 * $DragonFly: src/games/trek/nova.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  CAUSE A NOVA TO OCCUR
**
**      A nova occurs.  It is the result of having a star hit with
**      a photon torpedo.  There are several things which may happen.
**      The star may not be affected.  It may go nova.  It may turn
**      into a black hole.  Any (yummy) it may go supernova.
**
**      Stars that go nova cause stars which surround them to undergo
**      the same probabilistic process.  Klingons next to them are
**      destroyed.  And if the starship is next to it, it gets zapped.
**      If the zap is too much, it gets destroyed.
*/

void
nova(int x, int y)
{
        int             i, j;
        int             se;

        if (Sect[x][y] != STAR || Quad[Ship.quadx][Ship.quady].stars < 0)
                return;
        if (ranf(100) < 15)
```

```c
        {
                printf("Spock: Star at %d,%d failed to nova.\n", x, y);
                return;
        }
        if (ranf(100) < 5)
                return (snova(x, y));
        printf("Spock: Star at %d,%d gone nova\n", x, y);

        if (ranf(4) != 0)
                Sect[x][y] = EMPTY;
        else
        {
                Sect[x][y] = HOLE;
                Quad[Ship.quadx][Ship.quady].holes += 1;
        }
        Quad[Ship.quadx][Ship.quady].stars -= 1;
        Game.kills += 1;
        for (i = x - 1; i <= x + 1; i++)
        {
                if (i < 0 || i >= NSECTS)
                        continue;
                for (j = y - 1; j <= y + 1; j++)
                {
                        if (j < 0 || j >= NSECTS)
                                continue;
                        se = Sect[i][j];
                        switch (se)
                        {

                          case EMPTY:
                          case HOLE:
                                break;

                          case KLINGON:
                                killk(i, j);
                                break;

                          case STAR:
                                nova(i, j);
                                break;

                          case INHABIT:
                                kills(i, j, -1);
                                break;

                          case BASE:
                                killb(i, j);
                                Game.killb += 1;
                                break;

                          case ENTERPRISE:
                          case QUEENE:
                                se = 2000;
                                if (Ship.shldup)
                                {
                                        if (Ship.shield >= se)
                                        {
                                                Ship.shield -= se;
                                                se = 0;
                                        }
                                        else
                                        {
                                                se -= Ship.shield;
                                                Ship.shield = 0;
                                        }
                                }
                                Ship.energy -= se;
                                if (Ship.energy <= 0)
                                        lose(L_SUICID);
                                break;

                          default:
                                printf("Unknown object %c at %d,%d destroyed\n",
                                        se, i, j);
                                Sect[i][j] = EMPTY;
                                break;
                        }
                }
        }
        return;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)out.c    8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/out.c,v 1.4 1999/11/30 03:49:52 billf Exp $
 * $DragonFly: src/games/trek/out.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  Announce Device Out
*/

void
out(int dev)
{
        struct device   *d;

        d = &Device[dev];
        printf("%s reports %s ", d->person, d->name);
        if (d->name[strlen(d->name) - 1] == 's')
                printf("are");
        else
                printf("is");
        printf(" damaged\n");
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)phaser.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/phaser.c,v 1.5.2.1 2000/07/20 10:35:07 kris Exp $
 * $DragonFly: src/games/trek/phaser.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/* factors for phaser hits; see description below */

# define        ALPHA           3.0             /* spread */
# define        BETA            3.0             /* franf() */
# define        GAMMA           0.30            /* cos(angle) */
# define        EPSILON         150.0           /* dist ** 2 */
# define        OMEGA           10.596          /* overall scaling factor */

/* OMEGA ~= 100 * (ALPHA + 1) * (BETA + 1) / (EPSILON + 1) */

/*
**  Phaser Control
**
**      There are up to NBANKS phaser banks which may be fired
**      simultaneously.  There are two modes, "manual" and
**      "automatic".  In manual mode, you specify exactly which
**      direction you want each bank to be aimed, the number
**      of units to fire, and the spread angle.  In automatic
**      mode, you give only the total number of units to fire.
**
**      The spread is specified as a number between zero and
**      one, with zero being minimum spread and one being maximum
```

```
**      spread.  You  will normally want zero spread, unless your
**      short range scanners are out, in which case you probably
**      don't know exactly where the Klingons are.  In that case,
**      you really don't have any choice except to specify a
**      fairly large spread.
**
**      Phasers spread slightly, even if you specify zero spread.
**
**      Uses trace flag 30
*/

struct cvntab   Matab[] =
{
        { "m",          "anual",        (void (*)(int))1,       0 },
        { "a",          "utomatic",     (void (*)(int))0,       0 },
        { NULL,         NULL,           NULL,                   0 }
};

struct banks
{
        int     units;
        double  angle;
        double  spread;
};


void
phaser(__unused int unused)
{
        int             i;
        int                     j;
        struct kling    *k;
        double                  dx, dy;
        double                  anglefactor, distfactor;
        struct banks    *b;
        int                     manual, flag, extra = 0;
        int                     hit;
        double                  tot;
        int                     n;
        int                     hitreqd[NBANKS];
        struct banks            bank[NBANKS];
        struct cvntab           *ptr;

        if (Ship.cond == DOCKED) {
                printf("Phasers cannot fire through starbase shields\n");
                return;
        }
        if (damaged(PHASER)) {
                out(PHASER);
                return;
        }
        if (Ship.shldup) {
                printf("Sulu: Captain, we cannot fire through shields.\n");
                return;
        }
        if (Ship.cloaked)
        {
                printf("Sulu: Captain, surely you must realize that we cannot fire\n");
                printf("  phasers with the cloaking device up.\n");
                return;
        }
        /* decide if we want manual or automatic mode */
        manual = 0;
        if (testnl())
        {
                if (damaged(COMPUTER))
                {
                        printf("%s", Device[COMPUTER].name);
                        manual++;
                }
                else
                        if (damaged(SRSCAN))
                        {
                                printf("%s", Device[SRSCAN].name);
                                manual++;
                        }
                if (manual)
                        printf(" damaged, manual mode selected\n");
        }

        if (!manual)
        {
                ptr = getcodpar("Manual or automatic", Matab);
                manual = (long) ptr->value;
        }
        if (!manual && damaged(COMPUTER))
        {
                printf("Computer damaged, manual selected\n");
                skiptonl(0);
                manual++;
        }

        /* initialize the bank[] array */
        flag = 1;
        for (i = 0; i < NBANKS; i++)
                bank[i].units = 0;
        if (manual)
        {
                /* collect manual mode statistics */
                while (flag)
                {
                        printf("%d units available\n", Ship.energy);
                        extra = 0;
                        flag = 0;
                        for (i = 0; i < NBANKS; i++)
                        {
                                b = &bank[i];
                                printf("\nBank %d:\n", i);
                                hit = getintpar("units");
                                if (hit < 0)
                                        return;
                                if (hit == 0)
                                        break;
                                extra += hit;
                                if (extra > Ship.energy)
                                {
                                        printf("available energy exceeded. ");
                                        skiptonl(0);
                                        flag++;
                                        break;
                                }
                                b->units = hit;
                                hit = getintpar("course");
```

```
                                if (hit < 0 || hit > 360)
                                        return;
                                b->angle = hit * 0.0174532925;
                                b->spread = getfltpar("spread");
                                if (b->spread < 0 || b->spread > 1)
                                        return;
                        }
                        Ship.energy -= extra;
                }
                extra = 0;
        }
        else
        {
                /* automatic distribution of power */
                if (Etc.nkling <= 0) {
                        printf("Sulu: But there are no Klingons in this quadrant\n");
                        return;
                }
                printf("Phasers locked on target.  ");
                while (flag)
                {
                        printf("%d units available\n", Ship.energy);
                        hit = getintpar("Units to fire");
                        if (hit <= 0)
                                return;
                        if (hit > Ship.energy)
                        {
                                printf("available energy exceeded. ");
                                skiptonl(0);
                                continue;
                        }
                        flag = 0;
                        Ship.energy -= hit;
                        extra = hit;
                        n = Etc.nkling;
                        if (n > NBANKS)
                                n = NBANKS;
                        tot = n * (n + 1) / 2;
                        for (i = 0; i < n; i++)
                        {
                                k = &Etc.klingon[i];
                                b = &bank[i];
                                distfactor = k->dist;
                                anglefactor = ALPHA * BETA * OMEGA / (distfactor
 * distfactor + EPSILON);
                                anglefactor *= GAMMA;
                                distfactor = k->power;
                                distfactor /= anglefactor;
                                hitreqd[i] = distfactor + 0.5;
                                dx = Ship.sectx - k->x;
                                dy = k->y - Ship.secty;
                                b->angle = atan2(dy, dx);
                                b->spread = 0.0;
                                b->units = ((n - i) / tot) * extra;
#                               ifdef xTRACE
                                if (Trace)
                                {
                                        printf("b%d hr%d u%d df%.2f af%.2f\n",
                                                i, hitreqd[i], b->units,
                                                distfactor, anglefactor);
                                }
#                               endif
```

```
                                extra -= b->units;
                                hit = b->units - hitreqd[i];
                                if (hit > 0)
                                {
                                        extra += hit;
                                        b->units -= hit;
                                }
                        }

                        /* give out any extra energy we might have around */
                        if (extra > 0)
                        {
                                for (i = 0; i < n; i++)
                                {
                                        b = &bank[i];
                                        hit = hitreqd[i] - b->units;
                                        if (hit <= 0)
                                                continue;
                                        if (hit >= extra)
                                        {
                                                b->units += extra;
                                                extra = 0;
                                                break;
                                        }
                                        b->units = hitreqd[i];
                                        extra -= hit;
                                }
                                if (extra > 0)
                                        printf("%d units overkill\n", extra);
                        }
                }
        }

#       ifdef xTRACE
        if (Trace)
        {
                for (i = 0; i < NBANKS; i++)
                {
                        b = &bank[i];
                        printf("b%d u%d", i, b->units);
                        if (b->units > 0)
                                printf(" a%.2f s%.2f\n", b->angle, b->spread);
                        else
                                printf("\n");
                }
        }
#       endif

        /* actually fire the shots */
        Move.free = 0;
        for (i = 0; i < NBANKS; i++)
        {
                b = &bank[i];
                if (b->units <= 0)
                {
                        continue;
                }
                printf("\nPhaser bank %d fires:\n", i);
                n = Etc.nkling;
                k = Etc.klingon;
                for (j = 0; j < n; j++)
                {
```

```
                        if (b->units <= 0)
                                break;
                        /*
                        ** The formula for hit is as follows:
                        **
                        **  zap = OMEGA * [(sigma + ALPHA) * (rho + BETA)]
                        **      / (dist ** 2 + EPSILON)]
                        **      * [cos(delta * sigma) + GAMMA]
                        **      * hit
                        **
                        ** where sigma is the spread factor,
                        ** rho is a random number (0 -> 1),
                        ** GAMMA is a crud factor for angle (essentially
                        **      cruds up the spread factor),
                        ** delta is the difference in radians between the
                        **      angle you are shooting at and the actual
                        **      angle of the klingon,
                        ** ALPHA scales down the significance of sigma,
                        ** BETA scales down the significance of rho,
                        ** OMEGA is the magic number which makes everything
                        **      up to "* hit" between zero and one,
                        ** dist is the distance to the klingon
                        ** hit is the number of units in the bank, and
                        ** zap is the amount of the actual hit.
                        **
                        ** Everything up through dist squared should maximize
                        ** at 1.0, so that the distance factor is never
                        ** greater than one.  Conveniently, cos() is
                        ** never greater than one, but the same restric-
                        ** tion applies.
                        */
                        distfactor = BETA + franf();
                        distfactor *= ALPHA + b->spread;
                        distfactor *= OMEGA;
                        anglefactor = k->dist;
                        distfactor /= anglefactor * anglefactor + EPSILON;
                        distfactor *= b->units;
                        dx = Ship.sectx - k->x;
                        dy = k->y - Ship.secty;
                        anglefactor = atan2(dy, dx) - b->angle;
                        anglefactor = cos((anglefactor * b->spread) + GAMMA);
                        if (anglefactor < 0.0)
                        {
                                k++;
                                continue;
                        }
                        hit = anglefactor * distfactor + 0.5;
                        k->power -= hit;
                        printf("%d unit hit on Klingon", hit);
                        if (!damaged(SRSCAN))
                                printf(" at %d,%d", k->x, k->y);
                        printf("\n");
                        b->units -= hit;
                        if (k->power <= 0)
                        {
                                killk(k->x, k->y);
                                continue;
                        }
                        k++;
                }
        }
        /* compute overkill */
        for (i = 0; i < NBANKS; i++)
                extra += bank[i].units;
        if (extra > 0)
                printf("\n%d units expended on empty space\n", extra);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)play.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/play.c,v 1.2 1999/11/30 03:49:52 billf Exp $
 * $DragonFly: src/games/trek/play.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

static void     myreset(int);
/*
**  INSTRUCTION READ AND MAIN PLAY LOOP
**
**      Well folks, this is it.  Here we have the guts of the game.
**      This routine executes moves.  It sets up per-move variables,
**      gets the command, and executes the command.  After the command,
**      it calls events() to use up time, attack() to have Klingons
**      attack if the move was not free, and checkcond() to check up
**      on how we are doing after the move.
*/

struct cvntab   Comtab[] =
{
        { "abandon",            "",                     abandon,        0 },
        { "ca",                 "pture",                capture,        0 },
        { "cl",                 "oak",                  shield,         -1 },,
        { "c",                  "omputer",              computer,       0 },
        { "da",                 "mages",                dcrept,         0 },
        { "destruct",           "",                     destruct,       0 },
        { "do",                 "ck",                   dock,           0 },
        { "help",               "",                     help,           0 },
```

```
        { "i",                  "mpulse",               impulse,        0 },
        { "l",                  "rscan",                lrscan,         0 },
        { "m",                  "ove",                  dowarp,         0 },
        { "p",                  "hasers",               phaser,         0 },
        { "ram",                "",                     dowarp,         1 },
        { "dump",               "",                     dumpgame,       0 },
        { "r",                  "est",                  rest,           0 },
        { "sh",                 "ield",                 shield,         0 },
        { "s",                  "rscan",                srscan,         0 },
        { "st",                 "atus",                 srscan,         -1 },
        { "terminate",          "",                     myreset,        0 },
        { "t",                  "orpedo",               torped,         0 },
        { "u",                  "ndock",                undock,         0 },
        { "v",                  "isual",                visual,         0 },
        { "w",                  "arp",                  setwarp,        0 },
        { NULL,                 NULL,                   NULL,           0 }
};

static void
myreset(__unused int unused)
{
        longjmp(env, 1);
}

void
play(void)
{
        struct cvntab           *r;

        while (1)
        {
                Move.free = 1;
                Move.time = 0.0;
                Move.shldchg = 0;
                Move.newquad = 0;
                Move.resting = 0;
                skiptonl(0);
                r = getcodpar("\nCommand", Comtab);
                (*r->value)(r->value2);
                events(0);
                attack(0);
                checkcond();
        }
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)ram.c    8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/ram.c,v 1.4 1999/11/30 03:49:53 billf Exp $
 * $DragonFly: src/games/trek/ram.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  RAM SOME OBJECT
**
**      You have run into some sort of object.  It may be a Klingon,
**      a star, or a starbase.  If you run into a star, you are really
**      stupid, because there is no hope for you.
**
**      If you run into something else, you destroy that object.  You
**      also rack up incredible damages.
*/

void
ram(int ix, int iy)
{
        int             i;
        char            c;

        printf("\07RED ALERT\07: collision imminent\n");
        c = Sect[ix][iy];
        switch (c)
        {

          case KLINGON:
```

```
                printf("%s rams Klingon at %d,%d\n", Ship.shipname, ix, iy);
                killk(ix, iy);
                break;

          case STAR:
          case INHABIT:
                printf("Yeoman Rand: Captain, isn't it getting hot in here?\n");
                sleep(2);
                printf("Spock: Hull temperature approaching 550 Degrees Kelvin.\n");
                lose(L_STAR);

          case BASE:
                printf("You ran into the starbase at %d,%d\n", ix, iy);
                killb(Ship.quadx, Ship.quady);
                /* don't penalize the captain if it wasn't his fault */
                if (!damaged(SINS))
                        Game.killb += 1;
                break;
        }
        sleep(2);
        printf("%s heavily damaged\n", Ship.shipname);

        /* select the number of deaths to occur */
        i = 10 + ranf(20 * Game.skill);
        Game.deaths += i;
        Ship.crew -= i;
        printf("McCoy: Take it easy Jim; we had %d casualties.\n", i);

        /* damage devices with an 80% probability */
        for (i = 0; i < NDEV; i++)
        {
                if (ranf(100) < 20)
                        continue;
                damage(i, (2.5 * (franf() + franf()) + 1.0) * Param.damfac[i]);
        }

        /* no chance that your shields remained up in all that */
        Ship.shldup = 0;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)ranf.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/ranf.c,v 1.3 1999/11/30 03:49:53 billf Exp $
 * $DragonFly: src/games/trek/ranf.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

#include        "trek.h"

int
ranf(int max)
{
        if (max <= 0)
                return (0);
        return (random() % max);
}

double
franf(void)
{
        double          t;
        t = random() & 077777;
        return (t / 32767.0);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)rest.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/rest.c,v 1.4 1999/11/30 03:49:53 billf Exp $
 * $DragonFly: src/games/trek/rest.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  REST FOR REPAIRS
**
**      You sit around and wait for repairs to happen.  Actually, you
**      sit around and wait for anything to happen.  I do want to point
**      out however, that Klingons are not as patient as you are, and
**      they tend to attack you while you are resting.
**
**      You can never rest through a long range tractor beam.
**
**      In events() you will be given an opportunity to cancel the
**      rest period if anything momentous happens.
*/

void
rest(__unused int unused)
{
        double                  t;
        int             percent;

        /* get the time to rest */
        t = getfltpar("How long");
```

```c
        if (t <= 0.0)
                return;
        percent = 100 * t / Now.time + 0.5;
        if (percent >= 70)
        {
                printf("Spock: That would take %d%% of our remaining time.\n",
                        percent);
                if (!getynpar("Are you really certain that is wise"))
                        return;
        }
        Move.time = t;

        /* boundary condition is the LRTB */
        t = Now.eventptr[E_LRTB]->date - Now.date;
        if (Ship.cond != DOCKED && Move.time > t)
                Move.time = t + 0.0001;
        Move.free = 0;
        Move.resting = 1;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)schedule.c       8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/schedule.c,v 1.4 1999/11/30 03:49:53 billf Exp $
 * $DragonFly: src/games/trek/schedule.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"

/*
**  SCHEDULE AN EVENT
**
**      An event of type 'type' is scheduled for time NOW + 'offset'
**      into the first available slot.  'x', 'y', and 'z' are
**      considered the attributes for this event.
**
**      The address of the slot is returned.
*/

struct event *
schedule(int type, double offset, char x, char y, char z)
{
        struct event    *e;
        int             i;
        double                  date;

        date = Now.date + offset;
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                if (e->evcode)
                        continue;
```

```
                        /* got a slot */
#                       ifdef xTRACE
                        if (Trace)
                                printf("schedule: type %d @ %.2f slot %d parm %d %d %d\n",
                                        type, date, i, x, y, z);
#                       endif
                        e->evcode = type;
                        e->date = date;
                        e->x = x;
                        e->y = y;
                        e->systemname = z;
                        Now.eventptr[type] = e;
                        return (e);
                }
        syserr("Cannot schedule event %d parm %d %d %d", type, x, y, z);
        /* NOTREACHED */
        return(NULL);
}


/*
**  RESCHEDULE AN EVENT
**
**      The event pointed to by 'e' is rescheduled to the current
**      time plus 'offset'.
*/

void
reschedule(struct event *e1, double offset)
{
        double                  date;
        struct event    *e;

        e = e1;

        date = Now.date + offset;
        e->date = date;
#       ifdef xTRACE
        if (Trace)
                printf("reschedule: type %d parm %d %d %d @ %.2f\n",
                        e->evcode, e->x, e->y, e->systemname, date);
#       endif
        return;
}


/*
**  UNSCHEDULE AN EVENT
**
**      The event at slot 'e' is deleted.
*/

void
unschedule(struct event *e1)
{
        struct event    *e;

        e = e1;

#       ifdef xTRACE
        if (Trace)
                printf("unschedule: type %d @ %.2f parm %d %d %d\n",
                        e−>evcode, e−>date, e−>x, e−>y, e−>systemname);
#       endif
        Now.eventptr[e−>evcode & E_EVENT] = 0;
        e−>date = 1e50;
        e−>evcode = 0;
        return;
}


/*
**  Abreviated schedule routine
**
**      Parameters are the event index and a factor for the time
**      figure.
*/

struct event *
xsched(int ev1, int factor, int x, int y, int z)
{
        int     ev;

        ev = ev1;
        return (schedule(ev, −Param.eventdly[ev] * Param.time * log(franf()) / f
actor, x, y, z));
}


/*
**  Simplified reschedule routine
**
**      Parameters are the event index, the initial date, and the
**      division factor.  Look at the code to see what really happens.
*/

void
xresched(struct event *e1, int ev1, int factor)
{
        int             ev;
        struct event    *e;

        ev = ev1;
        e = e1;
        reschedule(e, −Param.eventdly[ev] * Param.time * log(franf()) / factor);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)score.c  8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/score.c,v 1.4 1999/11/30 03:49:53 billf Exp $
 * $DragonFly: src/games/trek/score.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  PRINT OUT THE CURRENT SCORE
*/

long
score(void)
{
        int             u;
        int             t;
        long                    s;
        double                  r;

        printf("\n*** Your score:\n");
        s = t = Param.klingpwr / 4 * (u = Game.killk);
        if (t != 0)
                printf("%d Klingons killed\t\t\t%6d\n", u, t);
        r = Now.date - Param.date;
        if (r < 1.0)
                r = 1.0;
        r = Game.killk / r;
        s += (t = 400 * r);
        if (t != 0)
```

```c
                printf("Kill rate %.2f Klingons/stardate  \t%6d\n", r, t);
        r = Now.klings;
        r /= Game.killk + 1;
        s += (t = -400 * r);
        if (t != 0)
                printf("Penalty for %d klingons remaining\t%6d\n", Now.klings, t);
        if (Move.endgame > 0)
        {
                s += (t = 100 * (u = Game.skill));
                printf("Bonus for winning a %s%s game\t\t%6d\n", Skitab[u - 1].abrev, Ski
tab[u - 1].full, t);
        }
        if (Game.killed)
        {
                s -= 500;
                printf("Penalty for getting killed\t\t  -500\n");
        }
        s += (t = -100 * (u = Game.killb));
        if (t != 0)
                printf("%d starbases killed\t\t\t%6d\n", u, t);
        s += (t = -100 * (u = Game.helps));
        if (t != 0)
                printf("%d calls for help\t\t\t%6d\n", u, t);
        s += (t = -5 * (u = Game.kills));
        if (t != 0)
                printf("%d stars destroyed\t\t\t%6d\n", u, t);
        s += (t = -150 * (u = Game.killinhab));
        if (t != 0)
                printf("%d inhabited starsystems destroyed\t%6d\n", u, t);
        if (Ship.ship != ENTERPRISE)
        {
                s -= 200;
                printf("penalty for abandoning ship\t\t  -200\n");
        }
        s += (t = 3 * (u = Game.captives));
        if (t != 0)
                printf("%d Klingons captured\t\t\t%6d\n", u, t);
        s += (t = -(u = Game.deaths));
        if (t != 0)
                printf("%d casualties\t\t\t\t%6d\n", u, t);
        printf("\n***  TOTAL\t\t\t%14ld\n", s);
        return (s);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)setup.c  8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/setup.c,v 1.6 1999/11/30 03:49:54 billf Exp $
 * $DragonFly: src/games/trek/setup.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  INITIALIZE THE GAME
**
**      The length, skill, and password are read, and the game
**      is initialized.  It is far too difficult to describe all
**      that goes on in here, but it is all straight-line code;
**      give it a look.
**
**      Game restart and tournament games are handled here.
*/

struct cvntab   Lentab[] =
{
        { "s",          "hort",         (void (*)(int))1,       0 },
        { "m",          "edium",        (void (*)(int))2,       0 },
        { "l",          "ong",          (void (*)(int))4,       0 },
        { "restart",    "",             NULL,                   0 },
        { NULL,         NULL,           NULL,                   0 }
};

struct cvntab   Skitab[] =
{
```

```c
        { "n",          "ovice",        (void (*)(int))1,       0 },
        { "f",          "air",          (void (*)(int))2,       0 },
        { "g",          "ood",          (void (*)(int))3,       0 },
        { "e",          "xpert",        (void (*)(int))4,       0 },
        { "c",          "ommodore",     (void (*)(int))5,       0 },
        { "i",          "mpossible",    (void (*)(int))6,       0 },
        { NULL,         NULL,           NULL,                   0 }
};

void
setup(void)
{
        struct cvntab           *r;
        int             i, j;
        double                  f;
        int                     d;
        int                     klump;
        int                     ix, iy;
        struct quad     *q;
        struct event            *e;

        while (1)
        {
                r = getcodpar("What length game", Lentab);
                Game.length = (long) r->value;
                if (Game.length == 0)
                {
                        if (restartgame())
                                continue;
                        return;
                }
                break;
        }
        r = getcodpar("What skill game", Skitab);
        Game.skill = (long) r->value;
        Game.tourn = 0;
        getstrpar("Enter a password", Game.passwd, 14, 0);
        if (sequal(Game.passwd, "tournament"))
        {
                getstrpar("Enter tournament code", Game.passwd, 14, 0);
                Game.tourn = 1;
                d = 0;
                for (i = 0; Game.passwd[i]; i++)
                        d += Game.passwd[i] << i;
                srandom(d);
        }
        Param.bases = Now.bases = ranf(6 - Game.skill) + 2;
        if (Game.skill == 6)
                Param.bases = Now.bases = 1;
        Param.time = Now.time = 6.0 * Game.length + 2.0;
        i = Game.skill;
        j = Game.length;
        Param.klings = Now.klings = i * j * 3.5 * (franf() + 0.75);
        if (Param.klings < i * j * 5)
                Param.klings = Now.klings = i * j * 5;
        if (Param.klings <= i)          /* numerical overflow problems */
                Param.klings = Now.klings = 127;
        Param.energy = Ship.energy = 5000;
        Param.torped = Ship.torped = 10;
        Ship.ship = ENTERPRISE;
        Ship.shipname = "Enterprise";
        Param.shield = Ship.shield = 1500;
        Param.resource = Now.resource = Param.klings * Param.time;
        Param.reserves = Ship.reserves = (6 - Game.skill) * 2.0;
        Param.crew = Ship.crew = 387;
        Param.brigfree = Ship.brigfree = 400;
        Ship.shldup = 1;
        Ship.cond = GREEN;
        Ship.warp = 5.0;
        Ship.warp2 = 25.0;
        Ship.warp3 = 125.0;
        Ship.sinsbad = 0;
        Ship.cloaked = 0;
        Param.date = Now.date = (ranf(20) + 20) * 100;
        f = Game.skill;
        f = log(f + 0.5);
        for (i = 0; i < NDEV; i++)
                if (Device[i].name[0] == '*')
                        Param.damfac[i] = 0;
                else
                        Param.damfac[i] = f;
        /* these probabilities must sum to 1000 */
        Param.damprob[WARP] = 70;       /* warp drive            7.0% */
        Param.damprob[SRSCAN] = 110;    /* short range scanners 11.0% */
        Param.damprob[LRSCAN] = 110;    /* long range scanners  11.0% */
        Param.damprob[PHASER] = 125;    /* phasers              12.5% */
        Param.damprob[TORPED] = 125;    /* photon torpedoes     12.5% */
        Param.damprob[IMPULSE] = 75;    /* impulse engines       7.5% */
        Param.damprob[SHIELD] = 150;    /* shield control       15.0% */
        Param.damprob[COMPUTER] = 20;   /* computer              2.0% */
        Param.damprob[SSRADIO] = 35;    /* subspace radio        3.5% */
        Param.damprob[LIFESUP] = 30;    /* life support          3.0% */
        Param.damprob[SINS] = 20;       /* navigation system     2.0% */
        Param.damprob[CLOAK] = 50;      /* cloaking device       5.0% */
        Param.damprob[XPORTER] = 80;    /* transporter           8.0% */
        /* check to see that I didn't blow it */
        for (i = j = 0; i < NDEV; i++)
                j += Param.damprob[i];
        if (j != 1000)
                syserr("Device probabilities sum to %d", j);
        Param.dockfac = 0.5;
        Param.regenfac = (5 - Game.skill) * 0.05;
        if (Param.regenfac < 0.0)
                Param.regenfac = 0.0;
        Param.warptime = 10;
        Param.stopengy = 50;
        Param.shupengy = 40;
        i = Game.skill;
        Param.klingpwr = 100 + 150 * i;
        if (i >= 6)
                Param.klingpwr += 150;
        Param.phasfac = 0.8;
        Param.hitfac = 0.5;
        Param.klingcrew = 200;
        Param.srndrprob = 0.0035;
        Param.moveprob[KM_OB] = 45;
        Param.movefac[KM_OB] = .09;
        Param.moveprob[KM_OA] = 40;
        Param.movefac[KM_OA] = -0.05;
        Param.moveprob[KM_EB] = 40;
        Param.movefac[KM_EB] = 0.075;
        Param.moveprob[KM_EA] = 25 + 5 * Game.skill;
        Param.movefac[KM_EA] = -0.06 * Game.skill;
        Param.moveprob[KM_LB] = 0;
```

```
        Param.movefac[KM_LB] = 0.0;
        Param.moveprob[KM_LA] = 10 + 10 * Game.skill;
        Param.movefac[KM_LA] = 0.25;
        Param.eventdly[E_SNOVA] = 0.5;
        Param.eventdly[E_LRTB] = 25.0;
        Param.eventdly[E_KATSB] = 1.0;
        Param.eventdly[E_KDESB] = 3.0;
        Param.eventdly[E_ISSUE] = 1.0;
        Param.eventdly[E_SNAP] = 0.5;
        Param.eventdly[E_ENSLV] = 0.5;
        Param.eventdly[E_REPRO] = 2.0;
        Param.navigcrud[0] = 1.50;
        Param.navigcrud[1] = 0.75;
        Param.cloakenergy = 1000;
        Param.energylow = 1000;
        for (i = 0; i < MAXEVENTS; i++)
        {
                e = &Event[i];
                e->date = 1e50;
                e->evcode = 0;
        }
        xsched(E_SNOVA, 1, 0, 0, 0);
        xsched(E_LRTB, Param.klings, 0, 0, 0);
        xsched(E_KATSB, 1, 0, 0, 0);
        xsched(E_ISSUE, 1, 0, 0, 0);
        xsched(E_SNAP, 1, 0, 0, 0);
        Ship.sectx = ranf(NSECTS);
        Ship.secty = ranf(NSECTS);
        Game.killk = Game.kills = Game.killb = 0;
        Game.deaths = Game.negenbar = 0;
        Game.captives = 0;
        Game.killinhab = 0;
        Game.helps = 0;
        Game.killed = 0;
        Game.snap = 0;
        Move.endgame = 0;

        /* setup stars */
        for (i = 0; i < NQUADS; i++)
                for (j = 0; j < NQUADS; j++)
                {
                        q = &Quad[i][j];
                        q->klings = q->bases = 0;
                        q->scanned = -1;
                        q->stars = ranf(9) + 1;
                        q->holes = ranf(3) - q->stars / 5;
                        q->qsystemname = 0;
                }

        /* select inhabited starsystems */
        for (d = 1; d < NINHAB; d++)
        {
                do
                {
                        i = ranf(NQUADS);
                        j = ranf(NQUADS);
                        q = &Quad[i][j];
                } while (q->qsystemname);
                q->qsystemname = d;
        }

        /* position starbases */
        for (i = 0; i < Param.bases; i++)
        {
                while (1)
                {
                        ix = ranf(NQUADS);
                        iy = ranf(NQUADS);
                        q = &Quad[ix][iy];
                        if (q->bases > 0)
                                continue;
                        break;
                }
                q->bases = 1;
                Now.base[i].x = ix;
                Now.base[i].y = iy;
                q->scanned = 1001;
                /* start the Enterprise near starbase */
                if (i == 0)
                {
                        Ship.quadx = ix;
                        Ship.quady = iy;
                }
        }

        /* position klingons */
        for (i = Param.klings; i > 0; )
        {
                klump = ranf(4) + 1;
                if (klump > i)
                        klump = i;
                while (1)
                {
                        ix = ranf(NQUADS);
                        iy = ranf(NQUADS);
                        q = &Quad[ix][iy];
                        if (q->klings + klump > MAXKLQUAD)
                                continue;
                        q->klings += klump;
                        i -= klump;
                        break;
                }
        }

        /* initialize this quadrant */
        printf("%d Klingons\n%d starbase", Param.klings, Param.bases);
        if (Param.bases > 1)
                printf("s");
        printf(" at %d,%d", Now.base[0].x, Now.base[0].y);
        for (i = 1; i < Param.bases; i++)
                printf(", %d,%d", Now.base[i].x, Now.base[i].y);
        printf("\nIt takes %d units to kill a Klingon\n", Param.klingpwr);
        Move.free = 0;
        initquad(0);
        srscan(1);
        attack(0);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)setwarp.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/setwarp.c,v 1.2 1999/11/30 03:49:54 billf Exp $
 * $DragonFly: src/games/trek/setwarp.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  SET WARP FACTOR
**
**      The warp factor is set for future move commands.  It is
**      checked for consistancy.
*/

void
setwarp(__unused int unused)
{
        double  warpfac;

        warpfac = getfltpar("Warp factor");
        if (warpfac < 0.0)
                return;
        if (warpfac < 1.0) {
                printf("Minimum warp speed is 1.0\n");
                return;
        }
        if (warpfac > 10.0) {
                printf("Maximum speed is warp 10.0\n");
                return;
```

```
        }
        if (warpfac > 6.0)
                printf("Damage to warp engines may occur above warp 6.0\n");
        Ship.warp = warpfac;
        Ship.warp2 = Ship.warp * warpfac;
        Ship.warp3 = Ship.warp2 * warpfac;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)shield.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/shield.c,v 1.5 1999/11/30 03:49:54 billf Exp $
 * $DragonFly: src/games/trek/shield.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  SHIELD AND CLOAKING DEVICE CONTROL
**
**      'f' is one for auto shield up (in case of Condition RED),
**      zero for shield control, and negative one for cloaking
**      device control.
**
**      Called with an 'up' or 'down' on the same line, it puts
**      the shields/cloak into the specified mode.  Otherwise it
**      reports to the user the current mode, and asks if she wishes
**      to change.
**
**      This is not a free move.  Hits that occur as a result of
**      this move appear as though the shields are half up/down,
**      so you get partial hits.
*/

struct cvntab Udtab[] =
{
        { "u",          "p",            (void (*)(int))1,       0 },
        { "d",          "own",          (void (*)(int))0,       0 },
        { NULL,         NULL,           NULL,                   0 }
```

```
};

void
shield(int f)
{
        int             i;
        struct cvntab           *r;
        char                    s[100];
        const char              *device, *dev2, *dev3;
        int                     ind;
        char                    *stat;

        if (f > 0 && (Ship.shldup || damaged(SRSCAN)))
                return;
        if (f < 0)
        {
                /* cloaking device */
                if (Ship.ship == QUEENE) {
                        printf("Ye Faire Queene does not have the cloaking device.\n");
                        return;
                }
                device = "Cloaking device";
                dev2 = "is";
                ind = CLOAK;
                dev3 = "it";
                stat = &Ship.cloaked;
        }
        else
        {
                /* shields */
                device = "Shields";
                dev2 = "are";
                dev3 = "them";
                ind = SHIELD;
                stat = &Ship.shldup;
        }
        if (damaged(ind))
        {
                if (f <= 0)
                        out(ind);
                return;
        }
        if (Ship.cond == DOCKED)
        {
                printf("%s %s down while docked\n", device, dev2);
                return;
        }
        if (f <= 0 && !testnl())
        {
                r = getcodpar("Up or down", Udtab);
                i = (long) r->value;
        }
        else
        {
                if (*stat)
                        sprintf(s, "%s %s up.  Do you want %s down", device, dev2, dev
3);
                else
                        sprintf(s, "%s %s down.  Do you want %s up", device, dev2, dev
3);
                if (!getynpar(s))
                        return;
                i = !*stat;
        }
        if (*stat == i)
        {
                printf("%s already ", device);
                if (i)
                        printf("up\n");
                else
                        printf("down\n");
                return;
        }
        if (i)
        {
                if (f >= 0)
                        Ship.energy -= Param.shupengy;
                else
                        Ship.cloakgood = 0;
        }
        Move.free = 0;
        if (f >= 0)
                Move.shldchg = 1;
        *stat = i;
        return;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)snova.c  8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/snova.c,v 1.4 1999/11/30 03:49:54 billf Exp $
 * $DragonFly: src/games/trek/snova.c,v 1.4 2007/05/13 22:25:41 swildner Exp $
 */

# include       "trek.h"

/*
**  CAUSE SUPERNOVA TO OCCUR
**
**      A supernova occurs.  If 'ix' < 0, a random quadrant is chosen;
**      otherwise, the current quadrant is taken, and (ix, iy) give
**      the sector quadrants of the star which is blowing up.
**
**      If the supernova turns out to be in the quadrant you are in,
**      you go into "emergency override mode", which tries to get you
**      out of the quadrant as fast as possible.  However, if you
**      don't have enough fuel, or if you by chance run into something,
**      or some such thing, you blow up anyway.  Oh yeh, if you are
**      within two sectors of the star, there is nothing that can
**      be done for you.
**
**      When a star has gone supernova, the quadrant becomes uninhab-
**      itable for the rest of eternity, i.e., the game.  If you ever
**      try stopping in such a quadrant, you will go into emergency
**      override mode.
*/

void
snova(int x, int y)
```

```
{
        int                     qx, qy;
        int             ix, iy = 0;
        int                     f;
        int                     dx, dy;
        int                     n;
        struct quad     *q;

        f = 0;
        ix = x;
        if (ix < 0)
        {
                /* choose a quadrant */
                while (1)
                {
                        qx = ranf(NQUADS);
                        qy = ranf(NQUADS);
                        q = &Quad[qx][qy];
                        if (q->stars > 0)
                                break;
                }
                if (Ship.quadx == qx && Ship.quady == qy)
                {
                        /* select a particular star */
                        n = ranf(q->stars);
                        for (ix = 0; ix < NSECTS; ix++)
                        {
                                for (iy = 0; iy < NSECTS; iy++)
                                        if (Sect[ix][iy] == STAR || Sect[ix][iy]
 == INHABIT)
                                                if ((n -= 1) <= 0)
                                                        break;
                                if (n <= 0)
                                        break;
                        }
                        f = 1;
                }
        }
        else
        {
                /* current quadrant */
                iy = y;
                qx = Ship.quadx;
                qy = Ship.quady;
                q = &Quad[qx][qy];
                f = 1;
        }
        if (f)
        {
                /* supernova is in same quadrant as Enterprise */
                printf("^G\nRED ALERT: supernova occurring at %d,%d\n", ix, iy);
                dx = ix - Ship.sectx;
                dy = iy - Ship.secty;
                if (dx * dx + dy * dy <= 2)
                {
                        printf("*** Emergency override attem");
                        sleep(1);
                        printf("\n");
                        lose(L_SNOVA);
                }
                q->scanned = 1000;
        }
	else
	{
		if (!damaged(SSRADIO))
		{
			q->scanned = 1000;
			printf("\nUhura: Captain, Starfleet Command reports a supernova\n");
			printf("  in quadrant %d,%d.  Caution is advised\n", qx, qy);
		}
	}

	/* clear out the supernova'ed quadrant */
	dx = q->klings;
	dy = q->stars;
	Now.klings -= dx;
	if (x >= 0)
	{
		/* Enterprise caused supernova */
		Game.kills += dy;
		if (q->bases)
			killb(qx, qy);
		Game.killk += dx;
	}
	else
		if (q->bases)
			killb(qx, qy);
	killd(qx, qy, (x >= 0));
	q->stars = -1;
	q->klings = 0;
	if (Now.klings <= 0)
	{
		printf("Lucky devil, that supernova destroyed the last klingon\n");
		win();
	}
	return;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)srscan.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/srscan.c,v 1.4 1999/11/30 03:49:55 billf Exp $
 * $DragonFly: src/games/trek/srscan.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  SHORT RANGE SENSOR SCAN
**
**      A short range scan is taken of the current quadrant.  If the
**      flag 'f' is one, it is an "auto srscan".  It does a status
**      report and a srscan.
**      If 'f' is -1, you get a status report only.  If it is zero,
**      you get a srscan and an optional status report.  The status
**      report is taken if you enter "srscan yes"; for all srscans
**      thereafter you get a status report with your srscan until
**      you type "srscan no".  It defaults to on.
**
**      The current quadrant is filled in on the computer chart.
*/

const char      *Color[4] =
{
        "GREEN",
        "DOCKED",
        "YELLOW",
        "RED"
};
```

```

void
srscan(int f)
{
        int             i, j;
        int             statinfo;
        const char              *s;
        int                     percent;
        struct quad             *q = NULL;
        struct cvntab           *p;

        if (f >= 0 && check_out(SRSCAN))
        {
                return;
        }
        if (f)
                statinfo = 1;
        else
        {
                if (!testnl())
                        Etc.statreport = getynpar("status report");
                statinfo = Etc.statreport;
        }
        if (f > 0)
                Etc.statreport = 1;
        if (f >= 0)
        {
                printf("\nShort range sensor scan\n");
                q = &Quad[Ship.quadx][Ship.quady];
                q->scanned = q->klings * 100 + q->bases * 10 + q->stars;
                printf(" ");
                for (i = 0; i < NSECTS; i++)
                {
                        printf("%d ", i);
                }
                printf("\n");
        }

        for (i = 0; i < NSECTS; i++)
        {
                if (f >= 0)
                {
                        printf("%d ", i);
                        for (j = 0; j < NSECTS; j++)
                                printf("%c ", Sect[i][j]);
                        printf("%d", i);
                        if (statinfo)
                                printf("  ");
                }
                if (statinfo)
                        switch (i)
                        {
                          case 0:
                                printf("stardate    %.2f", Now.date);
                                break;
                          case 1:
                                printf("condition   %s", Color[Ship.cond]);
                                if (Ship.cloaked)
                                        printf(", CLOAKED");
                                break;
                          case 2:
                                printf("position    %d,%d/%d,%d",Ship.quadx, Ship.qu
ady, Ship.sectx, Ship.secty);
                                break;
                        case 3:
                                printf("warp factor   %.1f", Ship.warp);
                                break;
                        case 4:
                                printf("total energy  %d", Ship.energy);
                                break;
                        case 5:
                                printf("torpedoes     %d", Ship.torped);
                                break;
                        case 6:
                                s = "down";
                                if (Ship.shldup)
                                        s = "up";
                                if (damaged(SHIELD))
                                        s = "damaged";
                                percent = 100.0 * Ship.shield / Param.shield;
                                printf("shields       %s, %d%%", s, percent);
                                break;
                        case 7:
                                printf("Klingons left %d", Now.klings);
                                break;
                        case 8:
                                printf("time left     %.2f", Now.time);
                                break;
                        case 9:
                                printf("life support  ");
                                if (damaged(LIFESUP))
                                {
                                        printf("damaged, reserves = %.2f", Ship.reserv
es);

                                        break;
                                }
                                printf("active");
                                break;
                        }
                printf("\n");
        }
        if (f < 0)
        {
                printf("current crew  %d\n", Ship.crew);
                printf("brig space    %d\n", Ship.brigfree);
                printf("Klingon power %d\n", Param.klingpwr);
                p = &Lentab[Game.length - 1];
                if (Game.length > 2)
                        p--;
                printf("Length, Skill %s%s, ", p->abrev, p->full);
                p = &Skitab[Game.skill - 1];
                printf("%s%s\n", p->abrev, p->full);
                return;
        }
        printf("  ");
        for (i = 0; i < NSECTS; i++)
                printf("%d ", i);
        printf("\n");

        if (q->qsystemname & Q_DISTRESSED)
                printf("Distressed ");
        if (q->qsystemname)
                printf("Starsystem %s\n", systemname(q));
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)systemname.c     8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/systemname.c,v 1.4 1999/11/30 03:49:55 billf Exp $
 * $DragonFly: src/games/trek/systemname.c,v 1.3 2006/09/07 21:19:44 pavalos Exp
 $
 */

# include       "trek.h"

/*
**  RETRIEVE THE STARSYSTEM NAME
**
**      Very straightforward, this routine just gets the starsystem
**      name.  It returns zero if none in the specified quadrant
**      (which, by the way, is passed it).
**
**      This routine knows all about such things as distressed
**      starsystems, etc.
*/

const char *
systemname(struct quad *q1)
{
        struct quad     *q;
        int             i;

        q = q1;

        i = q->qsystemname;
        if (i & Q_DISTRESSED)
                i = Event[i & Q_SYSTEM].systemname;
```

```

        i &= Q_SYSTEM;
        if (i == 0)
                return (0);
        return (Systemname[i]);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)torped.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/torped.c,v 1.5 1999/11/30 03:49:55 billf Exp $
 * $DragonFly: src/games/trek/torped.c,v 1.3 2006/09/07 21:19:44 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

static int      randcourse(int);

/*
**  PHOTON TORPEDO CONTROL
**
**      Either one or three photon torpedoes are fired.  If three
**      are fired, it is called a "burst" and you also specify
**      a spread angle.
**
**      Torpedoes are never 100% accurate.  There is always a random
**      cludge factor in their course which is increased if you have
**      your shields up.  Hence, you will find that they are more
**      accurate at close range.  However, they have the advantage that
**      at long range they don't lose any of their power as phasers
**      do, i.e., a hit is a hit is a hit, by any other name.
**
**      When the course spreads too much, you get a misfire, and the
**      course is randomized even more.  You also have the chance that
**      the misfire damages your torpedo tubes.
*/

void
```

```
torped(__unused int unused)
{
        int             ix, iy;
        double                  x, y, dx, dy;
        double                  angle;
        int                     course, course2;
        int             k;
        double                  bigger;
        double                  sectsize;
        int                     burst;
        int                     n;

        if (Ship.cloaked)
        {
                printf("Federation regulations do not permit attack while cloaked.\n");
                return;
        }
        if (check_out(TORPED))
                return;
        if (Ship.torped <= 0)
        {
                printf("All photon torpedos expended\n");
                return;
        }

        /* get the course */
        course = getintpar("Torpedo course");
        if (course < 0 || course > 360)
                return;
        burst = -1;

        /* need at least three torpedoes for a burst */
        if (Ship.torped < 3)
        {
                printf("No-burst mode selected\n");
                burst = 0;
        }
        else
        {
                /* see if the user wants one */
                if (!testnl())
                {
                        k = ungetc(cgetc(0), stdin);
                        if (k >= '0' && k <= '9')
                                burst = 1;
                }
        }
        if (burst < 0)
        {
                burst = getynpar("Do you want a burst");
        }
        if (burst)
        {
                burst = getintpar("burst angle");
                if (burst <= 0)
                        return;
                if (burst > 15) {
                        printf("Maximum burst angle is 15 degrees\n");
                        return;
                }
        }
        sectsize = NSECTS;
        n = -1;
        if (burst)
        {
                n = 1;
                course -= burst;
        }
        for (; n && n <= 3; n++)
        {
                /* select a nice random course */
                course2 = course + randcourse(n);
                angle = course2 * 0.0174532925;                 /* convert to ra
dians */
                dx = -cos(angle);
                dy =  sin(angle);
                bigger = fabs(dx);
                x = fabs(dy);
                if (x > bigger)
                        bigger = x;
                dx /= bigger;
                dy /= bigger;
                x = Ship.sectx + 0.5;
                y = Ship.secty + 0.5;
                if (Ship.cond != DOCKED)
                        Ship.torped -= 1;
                printf("Torpedo track");
                if (n > 0)
                        printf(", torpedo number %d", n);
                printf(":\n%6.1f\t%4.1f\n", x, y);
                while (1)
                {
                        ix = x += dx;
                        iy = y += dy;
                        if (x < 0.0 || x >= sectsize || y < 0.0 || y >= sectsize
)
                        {
                                printf("Torpedo missed\n");
                                break;
                        }
                        printf("%6.1f\t%4.1f\n", x, y);
                        switch (Sect[ix][iy])
                        {
                          case EMPTY:
                                continue;

                          case HOLE:
                                printf("Torpedo disappears into a black hole\n");
                                break;

                          case KLINGON:
                                for (k = 0; k < Etc.nkling; k++)
                                {
                                        if (Etc.klingon[k].x != ix || Etc.klingo
n[k].y != iy)
                                                continue;
                                        Etc.klingon[k].power -= 500 + ranf(501);
                                        if (Etc.klingon[k].power > 0)
                                        {
                                                printf("*** Hit on Klingon at %d,%d: exte
nsive damages\n",
                                                        ix, iy);
                                                break;
                                        }
```

```
                                        killk(ix, iy);
                                        break;
                                }
                                break;

                          case STAR:
                                nova(ix, iy);
                                break;

                          case INHABIT:
                                kills(ix, iy, -1);
                                break;

                          case BASE:
                                killb(Ship.quadx, Ship.quady);
                                Game.killb += 1;
                                break;
                          default:
                                printf("Unknown object %c at %d,%d destroyed\n",
                                        Sect[ix][iy], ix, iy);
                                Sect[ix][iy] = EMPTY;
                                break;
                        }
                        break;
                }
                if (damaged(TORPED) || Quad[Ship.quadx][Ship.quady].stars < 0)
                        break;
                course += burst;
        }
        Move.free = 0;
}


/*
**  RANDOMIZE COURSE
**
**      This routine randomizes the course for torpedo number 'n'.
**      Other things handled by this routine are misfires, damages
**      to the tubes, etc.
*/

static int
randcourse(int n)
{
        double                  r;
        int             d;

        d = ((franf() + franf()) - 1.0) * 20;
        if (abs(d) > 12)
        {
                printf("Photon tubes misfire");
                if (n < 0)
                        printf("\n");
                else
                        printf(" on torpedo %d\n", n);
                if (ranf(2))
                {
                        damage(TORPED, 0.2 * abs(d) * (franf() + 1.0));
                }
                d *= 1.0 + 2.0 * franf();
        }
        if (Ship.shldup || Ship.cond == DOCKED)
        {
                r = Ship.shield;
                r = 1.0 + r / Param.shield;
                if (Ship.cond == DOCKED)
                        r = 2.0;
                d *= r;
        }
        return (d);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)utility.c        8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/utility.c,v 1.5 1999/11/30 03:49:55 billf Exp $
 * $DragonFly: src/games/trek/utility.c,v 1.4 2006/09/07 21:19:45 pavalos Exp $
 */

#include <errno.h>
#include <stdarg.h>
#include "trek.h"

/*
**  ASSORTED UTILITY ROUTINES
*/

/*
**  BLOCK MOVE
**
**      Moves a block of storage of length 'l' bytes from the data
**      area pointed to by 'a' to the area pointed to by 'b'.
**      Returns the address of the byte following the 'b' field.
**      Overflow of 'b' is not tested.
*/

char *
bmove(const void *a, void *b, size_t l)
{
        return((char *)memcpy(b, a, l) + l);
}


/*
```

```c
**  STRING EQUALITY TEST
**      null-terminated strings 'a' and 'b' are tested for
**      absolute equality.
**      returns one if equal, zero otherwise.
*/

bool
sequal(const char *a, const char *b)
{
        return(!strcmp(a, b));
}


/*
**  SYSTEM ERROR
*/

void
syserr(const char *fmt, ...)
{
        va_list ap;

        va_start(ap, fmt);
        printf("\n\07TREK SYSERR: ");
        vprintf(fmt, ap);
        printf("\n");
        if (errno)
                printf("\tsystem error %d\n", errno);
        va_end(ap);
        exit(1);
}
```

```c
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)visual.c 8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/visual.c,v 1.4 1999/11/30 03:49:56 billf Exp $
 * $DragonFly: src/games/trek/visual.c,v 1.3 2006/09/07 21:19:45 pavalos Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/*
**  VISUAL SCAN
**
**      A visual scan is made in a particular direction of three sectors
**      in the general direction specified.  This takes time, and
**      Klingons can attack you, so it should be done only when sensors
**      are out.
*/

/* This struct[] has the delta x, delta y for particular directions */
struct xy       Visdelta[11] =
{
        { -1,   -1 },
        { -1,    0 },
        { -1,    1 },
        {  0,    1 },
        {  1,    1 },
        {  1,    0 },
        {  1,   -1 },
        {  0,   -1 },
        { -1,   -1 },
        { -1,    0 },
```

```c
        { -1,    1 }
};

void
visual(__unused int unused)
{
        int             ix, iy;
        int                     co;
        struct xy       *v;

        co = getintpar("direction");
        if (co < 0 || co > 360)
                return;
        co = (co + 22) / 45;
        v = &Visdelta[co];
        ix = Ship.sectx + v->x;
        iy = Ship.secty + v->y;
        if (ix < 0 || ix >= NSECTS || iy < 0 || iy >= NSECTS)
                co = '?';
        else
                co = Sect[ix][iy];
        printf("%d,%d %c ", ix, iy, co);
        v++;
        ix = Ship.sectx + v->x;
        iy = Ship.secty + v->y;
        if (ix < 0 || ix >= NSECTS || iy < 0 || iy >= NSECTS)
                co = '?';
        else
                co = Sect[ix][iy];
        printf("%c ", co);
        v++;
        ix = Ship.sectx + v->x;
        iy = Ship.secty + v->y;
        if (ix < 0 || ix >= NSECTS || iy < 0 || iy >= NSECTS)
                co = '?';
        else
                co = Sect[ix][iy];
        printf("%c %d,%d\n", co, ix, iy);
        Move.time = 0.05;
        Move.free = 0;
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)warp.c   8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/warp.c,v 1.4 1999/11/30 03:49:56 billf Exp $
 * $DragonFly: src/games/trek/warp.c,v 1.4 2007/05/13 18:33:55 swildner Exp $
 */

# include       "getpar.h"
# include       "trek.h"

/*
**  MOVE UNDER WARP POWER
**
**      This is both the "move" and the "ram" commands, differing
**      only in the flag 'fl'.  It is also used for automatic
**      emergency override mode, when 'fl' is < 0 and 'c' and 'd'
**      are the course and distance to be moved.  If 'fl' >= 0,
**      the course and distance are asked of the captain.
**
**      The guts of this routine are in the routine move(), which
**      is shared with impulse().  Also, the working part of this
**      routine is very small; the rest is to handle the slight chance
**      that you may be moving at some riduculous speed.  In that
**      case, there is code to handle time warps, etc.
*/

void
warp(int fl, int c, double d)
{
        int                     course;
        double                  power;
        double                  dist;
```

```
        double                  p_time;
        double                  speed;
        double                  frac;
        int             percent;
        int             i;
        char            *s;

        if (Ship.cond == DOCKED) {
                printf("%s is docked\n", Ship.shipname);
                return;
        }
        if (damaged(WARP))
        {
                out(WARP);
                return;
        }
        course = c;
        dist = d;

        /* check to see that we are not using an absurd amount of power */
        power = (dist + 0.05) * Ship.warp3;
        percent = 100 * power / Ship.energy + 0.5;
        if (percent >= 85)
        {
                printf("Scotty: That would consume %d%% of our remaining energy.\n",
                        percent);
                if (!getynpar("Are you sure that is wise"))
                        return;
        }

        /* compute the speed we will move at, and the time it will take */
        speed = Ship.warp2 / Param.warptime;
        p_time = dist / speed;

        /* check to see that that value is not ridiculous */
        percent = 100 * p_time / Now.time + 0.5;
        if (percent >= 85)
        {
                printf("Spock: That would take %d%% of our remaining time.\n",
                        percent);
                if (!getynpar("Are you sure that is wise"))
                        return;
        }

        /* compute how far we will go if we get damages */
        if (Ship.warp > 6.0 && ranf(100) < 20 + 15 * (Ship.warp - 6.0))
        {
                frac = franf();
                dist *= frac;
                p_time *= frac;
                damage(WARP, (frac + 1.0) * Ship.warp * (franf() + 0.25) * 0.20)
;
        }

        /* do the move */
        Move.time = move(fl, course, p_time, speed);

        /* see how far we actually went, and decrement energy appropriately */
        dist = Move.time * speed;
        Ship.energy -= dist * Ship.warp3 * (Ship.shldup + 1);

        /* test for bizarre events */
```

```c
        if (Ship.warp <= 9.0)
                return;
        printf("\n\n ___ Speed exceeding warp nine ___\n\n");
        sleep(2);
        printf("Ship’s safety systems malfunction\n");
        sleep(2);
        printf("Crew experiencing extreme sensory distortion\n");
        sleep(4);
        if (ranf(100) >= 100 * dist)
        {
                printf("Equilibrium restored −− all systems normal\n");
                return;
        }

        /* select a bizzare thing to happen to us */
        percent = ranf(100);
        if (percent < 70)
        {
                /* time warp */
                if (percent < 35 || !Game.snap)
                {
                        /* positive time warp */
                        p_time = (Ship.warp − 8.0) * dist * (franf() + 1.0);
                        Now.date += p_time;
                        printf("Positive time portal entered −− it is now Stardate %.2f\n",
                                Now.date);
                        for (i = 0; i < MAXEVENTS; i++)
                        {
                                percent = Event[i].evcode;
                                if (percent == E_FIXDV || percent == E_LRTB)
                                        Event[i].date += p_time;
                        }
                        return;
                }

                /* s/he got lucky: a negative time portal */
                p_time = Now.date;
                s = Etc.snapshot;
                bmove(s, Quad, sizeof Quad);
                bmove(s += sizeof Quad, Event, sizeof Event);
                bmove(s += sizeof Event, &Now, sizeof Now);
                printf("Negative time portal entered −− it is now Stardate %.2f\n",
                        Now.date);
                for (i = 0; i < MAXEVENTS; i++)
                        if (Event[i].evcode == E_FIXDV)
                                reschedule(&Event[i], Event[i].date − p_time);
                return;
        }

        /* test for just a lot of damage */
        if (percent < 80)
                lose(L_TOOFAST);
        printf("Equilibrium restored −− extreme damage occurred to ship systems\n");
        for (i = 0; i < NDEV; i++)
                damage(i, (3.0 * (franf() + franf()) + 1.0) * Param.damfac[i]);
        Ship.shldup = 0;
}

/*
 * dowarp() is used in a struct cvntab to call warp().  Since it is always ram
 * or move, fl is never < 0, so ask the user for course and distance, then pass
 * that to warp().
```

```c
 */
void
dowarp(int fl)
{
        int     c;
        double  d;

        if(getcodi(&c, &d))
                return;
        warp(fl, c, d);
}
```

```
/*
 * Copyright (c) 1980, 1993
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * @(#)win.c    8.1 (Berkeley) 5/31/93
 * $FreeBSD: src/games/trek/win.c,v 1.4 1999/11/30 03:49:56 billf Exp $
 * $DragonFly: src/games/trek/win.c,v 1.3 2006/09/07 21:19:45 pavalos Exp $
 */

# include       "trek.h"
# include       "getpar.h"

/*
**  Signal game won
**
**      This routine prints out the win message, arranges to print out
**      your score, tells you if you have a promotion coming to you,
**      cleans up the current input line, and arranges to have you
**      asked whether or not you want another game (via the longjmp()
**      call).
**
**      Pretty straightforward, although the promotion algorithm is
**      pretty off the wall.
*/

void
win(void)
{
        long                    s;
        struct cvntab   *p;

        sleep(1);
        printf("\nCongratulations, you have saved the Federation\n");
        Move.endgame = 1;
```

```

        /* print and return the score */
        s = score();

        /* decide if she gets a promotion */
        if (Game.helps == 0 && Game.killb == 0 && Game.killinhab == 0 && 5 * Gam
e.kills + Game.deaths < 100 &&
                        s >= 1000 && Ship.ship == ENTERPRISE)
        {
                printf("In fact, you are promoted one step in rank,\n");
                if (Game.skill >= 6)
                        printf("to the exalted rank of Commodore Emeritus\n");
                else
                {
                        p = &Skitab[Game.skill − 1];
                        printf("from %s%s ", p−>abrev, p−>full);
                        p++;
                        printf("to %s%s\n", p−>abrev, p−>full);
                }
        }

        /* clean out input, and request new game */
        skiptonl(0);
        longjmp(env, 1);
}
```