```
// Emacs style mode select   -*- C++ -*-
//-----------------------------------------------------------------------------
//
// $Id:$
//
// Copyright (C) 1993-1996 by id Software, Inc.
//
// This source is available for distribution and/or modification
// only under the terms of the DOOM Source Code License as
// published by id Software. All rights reserved.
//
// The source is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// FITNESS FOR A PARTICULAR PURPOSE. See the DOOM Source Code License
// for more details.
//
// $Log:$
//
// DESCRIPTION:
//      Teleportation.
//
//-----------------------------------------------------------------------------

static const char
rcsid[] = "$Id: p_telept.c,v 1.3 1997/01/28 22:08:29 b1 Exp $";



#include "doomdef.h"

#include "s_sound.h"

#include "p_local.h"


// Data.
#include "sounds.h"

// State.
#include "r_state.h"

```

```
42
43
44  //
45  // TELEPORTATION
46  //
47  int
48  EV_Teleport
49  ( line_t*       line,
50    int           side,
51    mobj_t*       thing )
52  {
53      int         i;
54      int         tag;
55      mobj_t*     m;
56      mobj_t*     fog;
57      unsigned    an;
58      thinker_t*  thinker;
59      sector_t*   sector;
60      fixed_t     oldx;
61      fixed_t     oldy;
62      fixed_t     oldz;
63
64      // don't teleport missiles
65      if (thing->flags & MF_MISSILE)
66          return 0;
67
68      // Don't teleport if hit back of line,
69      //  so you can get out of teleporter.
70      if (side == 1)
71          return 0;
72
73
74      tag = line->tag;
75      for (i = 0; i < numsectors; i++)
76      {
77          if (sectors[ i ].tag == tag )
78          {
79              thinker = thinkercap.next;
80              for (thinker = thinkercap.next;
81                   thinker != &thinkercap;
82                   thinker = thinker->next)
83              {
```

```
                // not a mobj
                if (thinker->function.acp1 != (actionf_p1)P_MobjThinker)
                    continue;

                m = (mobj_t *)thinker;

                // not a teleportman
                if (m->type != MT_TELEPORTMAN )
                    continue;

                sector = m->subsector->sector;
                // wrong sector
                if (sector-sectors != i )
                    continue;

                oldx = thing->x;
                oldy = thing->y;
                oldz = thing->z;

                if (!P_TeleportMove (thing, m->x, m->y))
                    return 0;

                thing->z = thing->floorz;  //fixme: not needed?
                if (thing->player)
                    thing->player->viewz = thing->z+thing->player->viewheight;

                // spawn teleport fog at source and destination
                fog = P_SpawnMobj (oldx, oldy, oldz, MT_TFOG);
                S_StartSound (fog, sfx_telept);
                an = m->angle >> ANGLETOFINESHIFT;
                fog = P_SpawnMobj (m->x+20*finecosine[an], m->y+20*finesine[an]
                                   , thing->z, MT_TFOG);

                // emit sound, where?
                S_StartSound (fog, sfx_telept);

                // don't move for a bit
                if (thing->player)
                    thing->reactiontime = 18;

                thing->angle = m->angle;
```

```
                thing->momx = thing->momy = thing->momz = 0;
                return 1;
            }
        }
    }
    return 0;
}


```