```c
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS for(int A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <SDL/SDL.h>
#include <SDL/SDL_mixer.h>

#include "wetspot2.h"
#include "font.h"
#include "palette.h"
#include "timer.h"
#include "logo.h"
#include "sprites.h"
#include "menu.h"
#include "world.h"
#include "input.h"
#include "sound.h"

// legacy data

// SDL compliant data
SDL_Surface *screen;
SDL_Surface *gamescreen;
SDL_Surface *theend;

Uint8 *keys;

GAMETYPE Game;

PLAYERTYPE Player[2];
ENEMYTYPE Enemy[MAXENEMIES];
OBJECTTYPE Object[MAXOBJS];
BLOCKTYPE Block[MAXBLOCKS];
SHOTTYPE Shot[MAXSHOTS];
DEATHTYPE Death[2];

CELL cell[12][20];

int Blocked;

// score
// Check if the player has gained enough points for an extra life
void CheckScore(int PlayerNum)
{
  if(Player[PlayerNum].score >= Player[PlayerNum].nextextra) {
  if(Player[PlayerNum].nextextra == 30000) Player[PlayerNum].nextextra = 0;
  Player[PlayerNum].nextextra += 100000;
  Player[PlayerNum].lives++;
  PlaySound(7);
  }
}
```

```c
// dX,dY: axis increase/decrease in each direction
// direction can be: 0 = right, 1 = up, 2 = left, 3 = down, 4 = nowhere
int dx[5] = {0, -1, 0, 1, 0};
int dy[5] = {1, 0, -1, 0, 0};

int GetFreeObject();
void KillEnemy(int);

// Checks game status and other minor game features
// Finds if all the players are dead; in that case, ends the game loop
void CheckStatus()
{
  if(Game.players == 1) {
    if(Player[0].dead == TRUE) {
      Game.status = 400;
      Player[0].dead = 2;
    }
  } else {
    if(Player[0].dead == TRUE) {
      Player[0].dead = 2;
      if(Player[0].dead >= 2 && Player[1].dead >= 2) Game.status = 400;
    }
    if(Player[1].dead == TRUE) {
      Player[1].dead = 2;
      if(Player[0].dead >= 2 && Player[1].dead >= 2) Game.status = 400;
    }
  }

  // Randomly adds a bonus object on the screen (if possible)
  if(Game.objects < 3 && Game.status != -501) {
    if(Game.status < 1 && Game.time > 15) {
      if(rand() % 100 == 0) { //INT(RND(1) * 100)
        int i = GetFreeObject();

        Object[i].x = rand() % 20; //INT(RND(1) * 20)
        Object[i].y = rand() % 12; //INT(RND(1) * 12)

        if(cell[Object[i].y][Object[i].x].st == 0) {
          Object[i].typ = 26 + rand() % 8; //INT(RND(1) * 8)
          Object[i].time = 0;

          if(rand() % 2 == 0) { //INT(RND(1) * 3)
            int xi = rand() % 20; //INT(RND(1) * 20):
            int yi = rand() % 12; //INT(RND(1) * 12)

            if(cell[yi][xi].rd > 0) Object[i].typ = cell[yi][xi].rd;
            if(Game.mode == DEMO && Object[i].typ == 14) Object[i].typ = 0;
          }

          if(rand() % 3 == 0) Object[i].typ = rand() % 33 + 1;
          Game.objects++;
        }
      }
    }
  }

  switch(Game.status) {
  case -500 ... -3:
    // The enemies are blocked by the clock
    Game.status++;
    if(Game.status == -2) {
      // Resumes the enemies
      Game.status = 0;
      if(Game.time < 16) {
        //ChangePal 0
      } else {
        //ChangePal -1
      }
    }
    break;
  case -2:
    // Do the lightnings
    Game.status = 0;
    for(int i = 0; i < 80; i++) {
      DrawScreen();
    }
    for(int i = 0; i < MAXENEMIES; i++) {
      if(Enemy[i].typ > 0) KillEnemy(i);
    }
    //ChangePal -1
    break;
  case -1:
    // Do the earthquake
    Game.status = 0;
    /*for(int i = 0 TO 100
      WAIT &H3DA, 8, 8: WAIT &H3DA, 8
      BlastCopy VARSEG(Buffer(0)), VARPTR(Buffer(0)), &HA000, ((i MOD 3) * 320)
    NEXT i*/
    for(int i = 0; i < MAXENEMIES; i++) {
      if(Enemy[i].typ > 0) KillEnemy(i);
    }
    break;
  default:
    if(Game.status > 0) Game.status++;
  }

  if(Game.mode == DEMO) return;

  // the game is paused
  if(keys[SDLK_RETURN]) {
    keys[SDLK_RETURN] = 0;
    TimerOn = FALSE;
    PlaySound(1);
    DrawScreen();
    SPrint("PAUSE!", 136, 96, Game.textcol);
    do {
      BlitAndWait(2);
    } while(!keys[SDLK_RETURN]);
    keys[SDLK_RETURN] = 0;
    if(Game.monsters > 0) TimerOn = TRUE;
  }
}

int Collide(int xPos, int yPos);

int GetFreeObject()
{
  for(int i = 0; i < MAXOBJS; i++)
    if(Object[i].typ == 0) return i;

  return (MAXOBJS-1);
}

// Moves shots and check collisions
void MoveShots()
{
  int Collision = -1;
  int mul = 0, expl = 0;

  for(int i = 0; i < MAXSHOTS; i++) {
    // There's a shot moving
    if(Shot[i].typ > 0) {
      switch(Shot[i].typ) {
      case 1:
      case 3:
        // 1: Bubble
        // 2: Egg
        // Moves the shot
        Shot[i].x += Shot[i].ax;
        Shot[i].y += Shot[i].ay;
        if(Shot[i].typ == 1) {
          mul = 2; expl = 4;
        } else {
          mul = 4; expl = 5;
        }
        // If the shot hits a block, it's destroyed
        if(Shot[i].x % 16 == 0 && Shot[i].y % 16 == 0) {
          if(cell[(Shot[i].y + (Shot[i].ay * mul)) / 16][(Shot[i].x + (Shot[i].ax * mul)) / 16].st > 0) {
            if(Shot[i].typ == 3) PlaySound(12);
            Shot[i].typ = expl;
            Shot[i].time = 0;
          }
        }
        Shot[i].time++;
        if(Shot[i].typ == 1) {
          Collision = Collide(Shot[i].x, Shot[i].y);
          if(Collision != -1) {
            // A player has been trapped by a bubble
            if(Player[Collision].status == 0) {
              Player[Collision].status = 120;
              Shot[i].typ = 0;
            }
          }
        } else {
          Collision = Collide(Shot[i].x, Shot[i].y);
          if(Collision != -1) {
            // A player has been hit by an egg; he's killed
            if(Player[Collision].status == 0) {
              Player[Collision].status = 201;
                Player[Collision].dir = -8;
                Player[Collision].speed = 2;
                Player[Collision].dead = TRUE;
                Player[Collision].lives--;
                if(Player[Collision].lives == -1) Player[Collision].levelreached = (Game.area * 5) +
Game.level;
                Shot[i].typ = 0;
                PlaySound(16);
              }
          }
        }
        break;
      case 2:
        // Green bouncing slime
        Shot[i].x += Shot[i].ax;
        Shot[i].y += Shot[i].ay;
        if(Shot[i].x % 16 == 0 && Shot[i].y % 16 == 0) {
          // If the shot hits a block, it bounces on it
          if(cell[(Shot[i].y + (Shot[i].ay * 4)) / 16][(Shot[i].x + (Shot[i].ax * 4)) / 16].st > 0) {
            Shot[i].ax = -Shot[i].ax;
            Shot[i].ay = -Shot[i].ay;
            PlaySound(6);
          }
        }
        Shot[i].time++;
        // The shot can't move forever...
        if(Shot[i].time > 200) Shot[i].typ = 0;
        Collision = Collide(Shot[i].x, Shot[i].y);
        if(Collision != -1) {
          if(Player[Collision].status == 0) {
            // A player has been hit by the shot; he's killed
            Player[Collision].status = 201;
            Player[Collision].dir = -8;
            Player[Collision].speed = 2;
            Player[Collision].dead = TRUE;
            Player[Collision].lives--;
            if(Player[Collision].lives == -1) Player[Collision].levelreached = (Game.area * 5) +
Game.level;
            Shot[i].typ = 0;
            PlaySound(16);
          }
        }
        break;
      case 4:
      case 5:
        // Each type of shot has its own explosion when it hits a block
        // Only the green slime doesn't have an explosion!
        Shot[i].time++;
        if(Shot[i].time == 6) Shot[i].typ = 0;
      }
    }
  }
}

// Handle potion bonus
void HandlePotion()
{
  int TotalBonus = 0, percent = 0, t;
  char str[64];

  for(int i = 0; i < MAXOBJS; i++) {
    if(Object[i].typ > 0) {
      // We have a bonus
      for(int ii = 0; ii < Game.players; ii++) {
        // Check if a player has got it
        if(Player[ii].x / 16 == Object[i].x && Player[ii].y / 16 == Object[i].y) {
          if(Player[ii].x % 16 == 0 && Player[ii].y % 16 == 0) {
            PlaySound(8);
            Object[i].typ = 0;
            Game.objects--;
            Player[ii].potion++;
            if(Game.objects == -1) {
              // No more bonuses on the screen
              TimerOn = FALSE;
              TotalBonus = Game.time * 10000;
              // Output special bonus statistics
              SPrint("POTION BONUS", 112, 50, Game.textcol);
              BlastLine(110, 59, 208, 59, Game.textcol);
              if(Game.players == 1) {
                // If we're in one player mode, show the PERFECT message only
                SPrint("PERFECT!", 128, 80, Game.textcol);
                sprintf(str, "%i PTS", TotalBonus);
                SPrint(str, 124, 96, Game.textcol);
                Player[0].score += TotalBonus;
              } else {
                for(int e = 0; e < 2; e++) {
                  // We're in two players mode
                  if(Player[e].dead == FALSE) {
                    sprintf(str, "PLAYER%i", e + 1);
                    SPrint(str, (80 + (e * 96)), 72, Game.textcol);
                    // Calculates percentage of bonuses taken by each player
                    percent = Player[e].potion / (MAXOBJS / 2) * 100;
                    if(percent == 0) {
                      // No bonuses taken by current player!
                      strcpy(str, "NO BONUS"); //s$ = "NO BONUS"
                      t = 0;
                    } else if(percent < 50) {
                      // If the player has got less than 50% of the bonuses, he
                      // gets half the total bonus.
                      strcpy(str, "HALF BONUS"); //s$ = "HALF BONUS"
                      t = -8;
                      Player[e].score += TotalBonus / 2;
                    } else {
                      // Otherwise he gains a PERFECT!
                      strcpy(str, "PERFECT!"); //s$ = "PERFECT!"
                      t = 0;
                      Player[e].score += TotalBonus;
                    }

                    char perc[64];
                    sprintf(perc, "%i%", percent); //perc$ = LTRIM$(STR$(percent)) + "%"
```

```
                        SPrint(perc, ((96 + ((4 - strlen(perc)) * 8)) + (e * 96)), 80, Game.textcol);
                        SPrint(str, (80 + (e * 96) + t), 88, Game.textcol);

                        strcpy(perc, "  0 PTS  "); //p$ = "  0 PTS  "
                        if(strcmp(str, "PERFECT!") == 0) {
                          sprintf(perc, "%i PTS", TotalBonus); //p$ = LTRIM$(STR$(TotalBonus)) + " PTS"
                        } else {
                          if(strcmp(str,"HALF BONUS") == 0)
                            sprintf(perc, "%i PTS", TotalBonus / 2); //p$ = LTRIM$(STR$(TotalBonus \ 2)) + "
PTS"
                        }
                        SPrint(perc, (76 + (e * 96)), 100, Game.textcol);
                      }
                    }
                  }
                  // Updates buffer and copies it on the screen
                  BlastLine(110, 115, 208, 115, Game.textcol);

                  for(int e = 0; e < 2; e++) {
                    CheckScore(e);
                  }

                  // Once the potion bonus is completed, the level is passed
                  Game.status = 501;
                }
              }
            }
          }
        }
      }
    }

// Checks for objects
void HandleObjects()
{
  char sbonus[] = "BONUS";

  for(int i = 0; i < MAXOBJS; i++) {
    if(Object[i].typ > 0) {
      // We've an object on the screen
      Object[i].time++;
      // If the object is on the screen for a long time, deletes it
      if(Object[i].time > 500) { Object[i].typ = 98; Object[i].time = 0; }

      switch(Object[i].typ) {
      case 97: // Enemy change effect
        if(Object[i].time == 12) Object[i].typ = 0;
        break;
      case 98: // The object is disappearing
        if(Object[i].time == 12) { Object[i].typ = 0; Game.objects--; }
        break;
      case 99: // Explosion
        if(Object[i].time == 9) Object[i].typ = 0;
        break;
      default: // Deletes score objects after a little time
        if(Object[i].typ > 99)
        if(Object[i].time == 40) { Object[i].typ = 0; Game.objects--; }
        break;
      }

      for(int ii = 0; ii < Game.players; ii++) {
        if(Player[ii].dead == FALSE) {
          // Checks is a player has got the object
          if(Player[ii].x / 16 == Object[i].x && Player[ii].y / 16 == Object[i].y) {
            if(Player[ii].x % 16 == 0 && Player[ii].y % 16 == 0) {
              int Points = 0;

              if(keys[SDLK_q]) goto _case_14;
              if(keys[SDLK_w]) goto _case_12;
              switch(Object[i].typ) {
```

```
        case 1 ... 5: // BONUS letters
          PlaySound(10);
          Player[ii].bonus[Object[i].typ-1] = sbonus[Object[i].typ-1];
          Points = 100;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          if(strcmp(Player[ii].bonus, "BONUS") == 0) {
            Player[ii].lives++;
            strcpy(Player[ii].bonus, "hhhhh");
            PlaySound(7);
          }
          break;
        case 6: // Invulnerability
          PlaySound(9);
          Points = 500;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          Player[ii].status = -600;
          break;
        case 7: // Player speed up
          PlaySound(9);
          Points = 500;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          Player[ii].speed = 4;
          break;
        case 8: // Earthquake shock
          Points = 500;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          Game.status = -1;
          TimerOn = FALSE;
          break;
        case 9: // Lightning shock
          Points = 500;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          //ChangePal 1;
          Game.status = -2;
          TimerOn = FALSE;
          break;
        case 10: // Stop clock
          PlaySound(3);
          Points = 500;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          //ChangePal 2;
          Game.status = -500;
          break;
        case 11: // Extra life
          PlaySound(7);
          Points = 1000;
          Object[i].typ = Points + (ii * 10000);
          Object[i].time = 0;
          Player[ii].lives++;
          break;
        case 12: // Level Warp
        _case_12:
          PlaySound(9);
          /*for(int e = 0; e < 5; e++)
            //PalSet 0, 63, 0, 0
            BlitAndWait(16);
            //PalSet 0, 0, 0, 0
            BlitAndWait(16);
          }*/
          Game.special = TRUE;
          Game.status = 500;
          break;
        case 13: // Dynamite
          Points = 100;
```

```
            Object[i].typ = Points + (ii * 10000);
            Object[i].time = 0;
            // Updates screen background
            for(int xi = 1; xi <= 18; xi++) {
              for(int yi = 1; yi <= 10; yi++) {
                if((((Object[i].x - xi) * (Object[i].x - xi)) + ((Object[i].y - yi) * (Object[i].y - 
yi))) < 17) {
                  if(cell[yi][xi].st == 2) {
                    int bonus = cell[yi][xi].rd;
                    PutShape(236 + cell[yi][xi].nd, xi * 16, yi * 16);
                    cell[yi][xi].st = 0;
                    cell[yi][xi].rd = 0;

                    if(cell[yi-1][xi].st > 0)
                      PutShape(232 + cell[yi-1][xi].st, xi * 16, yi * 16);

                    if(cell[yi+1][xi].st == 0)
                      PutShape(236 + cell[yi+1][xi].nd, xi * 16, (yi + 1) * 16);
                    int iii = GetFreeObject();
                    Object[iii].x = xi;
                    Object[iii].y = yi;
                    Object[iii].typ = 99;
                    Object[iii].time = 0;
                    if(bonus > 0) {
                      // If a destroyed block has a hidden object, shows it
                      iii = GetFreeObject();
                      Object[iii].x = xi;
                      Object[iii].y = yi;
                      Object[iii].typ = bonus;
                      Object[iii].time = 220;
                      Game.objects++;
                    }
                  }
                }
              }
            }
            break;
          case 14: // Potion bonus
          _case_14:
            PlaySound(9);
            TimerOn = FALSE;
            Game.status = -501;
            for(int e = 0; e < MAXENEMIES; e++) Enemy[e].typ = 0;
            for(int e = 0; e < MAXSHOTS; e++) Shot[e].typ = 0;

            //memset(Enemy, 0, sizeof(Enemy));
            //memset(Shot, 0, sizeof(Shot));

            // Changes all the moveable blocks into fixed ones and updates
            // the screen background
            for(int xi = 1; xi <= 18; xi++) {
              for(int yi = 1; yi <= 10; yi++) {
                if(cell[yi][xi].st == 2) {
                  PutShape(236 + cell[yi][xi].nd, xi * 16, yi * 16);
                  PutShape(235, xi * 16, yi * 16);

                  cell[yi][xi].st = 1;
                  cell[yi][xi].nd = 1;
                }
              }
            }

            // Chooses a random bonus item
            int bonus = 26 + rand() % 8; //INT(RND(1) * 8)
            for(int e = 0; e < MAXOBJS; e++) Object[e].typ = 0;
            //memset(Object, 0, sizeof(Object));
            int NumPotions = MAXOBJS;
            if(NumPotions > 23) NumPotions = 23;
            // Puts the special bonuses into the objects queue
            for(int e = 0; e < NumPotions; e++) {
```

```
        int xi = 1, yi = 1;
        // don't put srand() in cycle or rand() will give the same results
        srand(time(NULL));
        do {
          xi = rand() % 18 + 1; //INT(RND(1) * 18) + 1
          yi = rand() % 10 + 1; //INT(RND(1) * 10) + 1

          if(cell[yi][xi].st == 0 && cell[yi][xi].rd == 0) break;
        } while(1);

        cell[yi][xi].st = 0;
        cell[yi][xi].nd = 1;
        cell[yi][xi].rd = 1;
        Object[e].typ = bonus;
        Object[e].x = xi;
        Object[e].y = yi;
      }
      // Initializes potion bonus game mode variables
      Game.objects = NumPotions;
      Game.time = 20 - ((Game.players != 1) * 5);
      for(int e = 0; e < Game.players; e++) Player[e].potion = 0;
      Game.mode = POTIONBONUS;
      TimerOn = TRUE;
      return;
    case 15: // Present
      PlaySound(9);
      Points = 500;
      Object[i].typ = Points + (ii * 10000);
      Object[i].time = 0;
      Game.special = rand() % 16 + 18; //INT(RND(1) * 16) + 18
      break;
    case 16: // Extra points
      PlaySound(8);
      Points = 9000;
      Object[i].typ = Points + (ii * 10000);
      Object[i].time = 0;
      break;
    case 17: // Chest (????)
      srand(time(NULL));
      switch(rand() % 4) { //INT(RND(1) * 4)
      case 0: // Extra points
        PlaySound(8);
        Points = 5000;
        Object[i].typ = Points + (ii * 10000);
        Object[i].time = 0;
        break;
      case 1: // Player slow down
        PlaySound(9);
        Points = 100;
        Object[i].typ = Points + (ii * 10000);
        Object[i].time = 0;
        Player[ii].speed = 1;
        break;
      case 2: // Extra life
        PlaySound(7);
        Points = 500;
        Object[i].typ = Points + (ii * 10000);
        Object[i].time = 0;
        Player[ii].lives++;
        break;
      case 3: // Monster change
        PlaySound(9);
        Points = 500;
        Object[i].typ = Points + (ii * 10000);
        Object[i].time = 0;
        int ChangeTo = rand() % 5 + 1; //INT(RND(1) * 5) + 1
        if(ChangeTo == 3) ChangeTo = 7;
        if(ChangeTo == 5) ChangeTo = 6;
        for(int e = 0; e < MAXENEMIES; e++) {
          if(Enemy[e].typ > 0) Enemy[e].change = ChangeTo;
```

```
                }
              }
              break;
            case 18 ... 33: // Normal bonuses
              PlaySound(8);
              Points = (((Object[i].typ - 18) % 8) + 1) * 100;
              Object[i].typ = Points + (ii * 10000);
              Object[i].time = 0;
            }
            // If we're in normal game mode, adds the score
            if(Game.mode < POTIONBONUS) Player[ii].score += Points;
            CheckScore(ii);
            Game.objects--;
          }
        }
      }
    }
  }
}
}

// kill the enemy (it'll release a bonus!)
void KillEnemy(int EnemyNum)
{
  PlaySound(11);
  // Turn the enemy into a bouncing one
  Enemy[EnemyNum].typ = -Enemy[EnemyNum].typ;
  Enemy[EnemyNum].x = (Enemy[EnemyNum].x / 4) * 4;
  Enemy[EnemyNum].action = FALSE;
  Enemy[EnemyNum].change = 0;

  switch(rand() % 4) { //INT(RND(1) * 4)
  case 0: Enemy[EnemyNum].ox = -4; break;
  case 1: Enemy[EnemyNum].ox = -2; break;
  case 2: Enemy[EnemyNum].ox = 2; break;
  case 3: Enemy[EnemyNum].ox = 4; break;
  }

  Enemy[EnemyNum].oy = -12;
  Enemy[EnemyNum].z = 8;
  Enemy[EnemyNum].frame = 0;

  // Decreases monsters number
  Game.monsters--;
  if(Game.monsters == 0) {
    // No more enemies on the current level!
    TimerOn = FALSE;
    // Delete all the special bonuses from the screen
    for(int e = 0; e < MAXOBJS; e++) {
      if(Object[e].typ > 0 && Object[e].typ < 18) {
        Object[e].typ = 98;
        Object[e].time = 0;
      }
    }
    if(Game.special == FALSE) {
      // Allows the players to collect some bonuses for a little time before
      // ending the level
      Game.status = 200;
    } else {
      // The "present" bonus has been taken; puts lots of bonuses randomly on
      // the screen
      for(int e = 0; e < (MAXOBJS / 3) + 1; e++) {
        int xi, yi;
        srand(time(NULL));
        do {
          xi = rand() % 18 + 1; //INT(RND(1) * 18) + 1
          yi = rand() % 10 + 1; //INT(RND(1) * 10) + 1
          if(cell[yi][xi].st == 0 && cell[yi][xi].rd == 0) break;
        } while(1);
        //Cel(xi, yi) = (nd * 100) + 1
        cell[yi][xi].st = 0;
        cell[yi][xi].nd = 1;
        cell[yi][xi].rd = 1;

        Object[e].typ = Game.special;
        Object[e].time = 0;
        Object[e].x = xi;
        Object[e].y = yi;
      }
      // Gives the players more time to collect them
      Game.status = 1;
    }
  }
}

// Find which player is nearest to the given enemy position
int FindTarget(int xPos, int yPos)
{
  int result = 0, dist1, dist2;

  if(Game.players == 1) return 0; // 1 ??

  dist1 = ((Player[0].x / 16) - xPos) * ((Player[0].x / 16) - xPos) + ((Player[0].y / 16) - yPos) *
((Player[0].y / 16) - yPos);
  dist2 = ((Player[1].x / 16) - xPos) * ((Player[1].x / 16) - xPos) + ((Player[1].y / 16) - yPos) *
((Player[1].y / 16) - yPos);
  if(dist1 < dist2) result = 0; else result = 1;
  if(Player[result].dead == TRUE || Player[result].dead >= 2) result ^= 1;

  return result;
}

// Returns true if the specified enemy is in front of a player and the player
// is free
int InFrontOf(int EnemyNum, int PlayerNum)
{
  if(Player[PlayerNum].dead != FALSE) return FALSE;

  if(Enemy[EnemyNum].y == Player[PlayerNum].y) {
    if(Enemy[EnemyNum].x > Player[PlayerNum].x) {
      if(Enemy[EnemyNum].dir == 1) return TRUE;
    }
    if(Enemy[EnemyNum].x < Player[PlayerNum].x) {
      if(Enemy[EnemyNum].dir == 3) return TRUE;
    }
  }

  if(Enemy[EnemyNum].x == Player[PlayerNum].x) {
    if(Enemy[EnemyNum].y > Player[PlayerNum].y) {
      if(Enemy[EnemyNum].dir == 2) return TRUE;
    }
    if(Enemy[EnemyNum].y < Player[PlayerNum].y) {
      if(Enemy[EnemyNum].dir == 0) return TRUE;
    }
  }

  if (Player[PlayerNum].status != 0) return FALSE;

  return FALSE;
}

// Moves the unbeatable enemies and checks for them to hit players
void MoveDeath()
{
  if(Game.time > 0) return;
  if(Game.status < -500 || Game.status > 0) return;

  for(int i = 0; i < Game.players; i++) {
    // Move the flame
    if(Player[i].x < Death[i].x) Death[i].x -= Death[i].speed;
    if(Player[i].x > Death[i].x) Death[i].x += Death[i].speed;
    if(Player[i].y < Death[i].y) Death[i].y -= Death[i].speed;
    if(Player[i].y > Death[i].y) Death[i].y += Death[i].speed;
    Death[i].frame = ((Death[i].frame + 1) % 4);

    // Checks if it hits the player
    if(Player[i].dead == FALSE) {
      if(Death[i].x > Player[i].x - 12 && Death[i].x < Player[i].x + 12) {
        if(Death[i].y > Player[i].y - 12 && Death[i].y < Player[i].y + 12) {
          // Kills the player
          Player[i].status = 201;
          Player[i].dir = -8;
          Player[i].speed = 2;
          Player[i].dead = TRUE;
          Player[i].lives--;
          if(Player[i].lives == -1)
            Player[i].levelreached = (Game.area * 5) + Game.level;
          PlaySound(16);
        }
      }
    }
  }

}

// Process a sort of AI to move monsters on the level.
void MoveEnemies()
{
  int AI = 0, pd[4], wayback;

  if(Blocked == TRUE) return;

  srand(time(NULL));

  for(int i = 0; i < MAXENEMIES; i++) {
    // Are the monsters to be changed to a different type?
    if(Enemy[i].change > 0 && Game.status > -3) {
      if(Enemy[i].x % 16 == 0 && Enemy[i].y % 16 == 0) {
        Enemy[i].typ = Enemy[i].change;
        Enemy[i].change = 0;

        // Reinitialize the enemy variables according to its new type
        switch(Enemy[i].typ) {
        case 1:
          Enemy[i].z = rand() % 7; //INT(RND(1) * 7)
          Enemy[i].az = 1;
          break;
        case 4:
          Enemy[i].z = rand() % 10; //INT(RND(1) * 10)
          Enemy[i].az = 1;
          break;
        default:
          Enemy[i].z = 1;
          Enemy[i].az = 0;
        }

        Enemy[i].ox = Enemy[i].x / 16;
        Enemy[i].oy = Enemy[i].y / 16;
        Enemy[i].dir = 4;
        Enemy[i].frame = 0;
        Enemy[i].aframe = 1;
        Enemy[i].action = 0;

        int ii = GetFreeObject();
        Object[ii].typ = 97;
        Object[ii].time = 0;
        Object[ii].x = Enemy[i].x / 16;
        Object[ii].y = Enemy[i].y / 16;

      }
```

```
}

if(Enemy[i].typ > 0 && Game.status > -3) {
  // Change enemy's direction
  if(Enemy[i].x % 16 == 0 && Enemy[i].y % 16 == 0) {
    int OldDir = Enemy[i].dir;
    if(Enemy[i].dir == 4) Enemy[i].dir = rand() % 4; //INT(RND(1) * 4)

    // Process shark enemy
    if(Enemy[i].typ == 3) {
      int xc = Enemy[i].x / 16;
      int yc = Enemy[i].y / 16;

      for(int ii = 0; ii < 4; ii++) {
        switch(Enemy[i].dir) {
        case 0:
          if(cell[yc][xc-1].st > 0) Enemy[i].ox = 2;
          if(cell[yc+1][xc].st > 0) Enemy[i].oy = -2;
          if(cell[yc+1][xc-1].st > 0 && cell[yc][xc-1].st == 0 && cell[yc+1][xc].st == 0) {
            Enemy[i].ox = 2;
            Enemy[i].oy = -2;
          }
          break;
        case 1:
          if(cell[yc][xc-1].st > 0) Enemy[i].ox = 2;
          if(cell[yc-1][xc].st > 0) Enemy[i].oy = 2;
          if(cell[yc-1][xc-1].st > 0 && cell[yc][xc-1].st == 0 && cell[yc-1][xc].st == 0) {
            Enemy[i].ox = 2;
            Enemy[i].oy = 2;
          }
          break;
        case 2:
          if(cell[yc][xc+1].st > 0) Enemy[i].ox = -2;
          if(cell[yc-1][xc].st > 0) Enemy[i].oy = 2;
          if(cell[yc-1][xc+1].st > 0 && cell[yc][xc+1].st == 0 && cell[yc-1][xc].st == 0) {
            Enemy[i].ox = -2;
            Enemy[i].oy = 2;
          }
          break;
        case 3:
          if(cell[yc][xc+1].st > 0) Enemy[i].ox = -2;
          if(cell[yc+1][xc].st > 0) Enemy[i].oy = -2;
          if(cell[yc+1][xc+1].st > 0 && cell[yc][xc+1].st == 0 && cell[yc+1][xc].st == 0) {
            Enemy[i].ox = -2;
            Enemy[i].oy = -2;
          }
          break;
        }

        if(Enemy[i].ox < 0 && Enemy[i].oy > 0) Enemy[i].dir = 0;
        if(Enemy[i].ox < 0 && Enemy[i].oy < 0) Enemy[i].dir = 1;
        if(Enemy[i].ox > 0 && Enemy[i].oy < 0) Enemy[i].dir = 2;
        if(Enemy[i].ox > 0 && Enemy[i].oy > 0) Enemy[i].dir = 3;
      }

      if(cell[yc][xc-1].st > 0 &&
         cell[yc-1][xc].st > 0 &&
         cell[yc][xc+1].st > 0 &&
         cell[yc+1][xc].st > 0) {
        Enemy[i].dir = 4;
      }

      if(cell[yc-1][xc-1].st > 0 &&
         cell[yc-1][xc+1].st > 0 &&
         cell[yc+1][xc+1].st > 0 &&
         cell[yc+1][xc-1].st > 0) {
        Enemy[i].dir = 4;
      }

    // Process other enemies when it's time to change direction
```

```
} else {
  // Each enemy has its AI factor
  switch(Enemy[i].typ) {
  case 1: if(Enemy[i].action > 0) AI = 9; else AI = 4; break;
  case 2: AI = 2; break;
  case 4: AI = 5; break;
  case 5: AI = 5; break;
  case 6: AI = 6; break;
  case 7: AI = 6; break;
  }

  if(Game.time < 15 && AI < 5) AI = 5;
  wayback = 4;
  // Finds the status of each block surrounding the enemy
  for(int ii = 0; ii < 4; ii++) {
    //GetBlockInfo Cel((Enemy[i].x \ 16) + dx(ii), (Enemy[i].y \ 16) + dy(ii))
    if(cell[Enemy[i].y / 16 + dy[ii]][Enemy[i].x / 16 + dx[ii]].st == 0) {
      pd[ii] = 1;
      if(Enemy[i].ox == Enemy[i].x / 16 + dx[ii] &&
         Enemy[i].oy == Enemy[i].y / 16 + dy[ii]) {pd[ii] = 2; wayback = ii;}
    } else {
      pd[ii] = 0;
    }
  }

  // Try to choose a way
  int Found = -1, d = 0;
  srand(time(NULL));
  for(int ii = 0; ii < 16; ii++) {
    d = rand() % 4; //INT(RND(1) * 4)
    if(pd[d] == 1) { Found = d; break; }
  }

  // Way found!
  if(Found > -1) {
    Enemy[i].dir = d;
    // If the enemy is enough smart, it seeks the player
    if(rand() % 10 + 1 < AI) { // (INT(RND(1) * 10) + 1)
      int p = FindTarget(Enemy[i].x / 16, Enemy[i].y / 16);
      int ok = FALSE;
      if(Player[p].y > Enemy[i].y && pd[0] > 0) {
        Enemy[i].dir = 0;
        ok = TRUE;
      }
      if(!ok && Player[p].x < Enemy[i].x && pd[1] > 0)
        { Enemy[i].dir = 1; ok = TRUE; }
      if(!ok && Player[p].y < Enemy[i].y && pd[2] > 0)
        { Enemy[i].dir = 2; ok = TRUE; }
      if(!ok && Player[p].x > Enemy[i].x && pd[3] > 0)
        { Enemy[i].dir = 3; ok = TRUE; }
      if(!ok && Player[p].y > Enemy[i].y && pd[0] > 0)
        { Enemy[i].dir = 0; ok = TRUE; }
      if(!ok && Player[p].x < Enemy[i].x && pd[1] > 0)
        Enemy[i].dir = 1;
    }
  } else {
    // No way; the enemy tries to come back on its steps
    Enemy[i].dir = wayback;
    Enemy[i].action = 0;
  }

  // The enemy can't move in any direction!
  if(pd[0] == 0 && pd[1] == 0 && pd[2] == 0 && pd[3] == 0)
    { Enemy[i].dir = 4; Enemy[i].action = 0; }
}

// Special tasks
if(Enemy[i].dir != 4) {
  switch(Enemy[i].typ) {
  case 1:
        if(Enemy[i].action > 0) {
          Enemy[i].action--;
        } else {
          if(InFrontOf(i, FindTarget(Enemy[i].x / 16, Enemy[i].y / 16)) == TRUE)
            Enemy[i].action = rand() % 10 + 5; //INT(RND(1) * 10) + 5
        }
        break;
      case 2:
        if(Enemy[i].dir == 0 ||Enemy[i].dir == 3) Enemy[i].aframe = 1; else Enemy[i].aframe = -1;
        break;
      case 4:
      case 5:
      case 7:
        if(Enemy[i].action > 0) {
          Enemy[i].dir = OldDir;
          Enemy[i].action--;
          if(Enemy[i].action == 0) {
            for(int ii = 0; ii < MAXSHOTS; ii++)
              if(Shot[ii].typ == 0) {
                int s = 0; // 1 ??
                switch(Enemy[i].typ) {
                case 4: Shot[ii].typ = 1; s = 8; break;
                case 5: Shot[ii].typ = 2; s = 4; break;
                case 7: Shot[ii].typ = 3; s = 4; break;
                }
                Shot[ii].x = Enemy[i].x + dx[Enemy[i].dir] * s;
                Shot[ii].y = Enemy[i].y + dy[Enemy[i].dir] * s;
                Shot[ii].ax = dx[Enemy[i].dir] * s;
                Shot[ii].ay = dy[Enemy[i].dir] * s;
                Shot[ii].time = 0;
                break;
              }
          }
        } else {
          if(InFrontOf(i, FindTarget(Enemy[i].x / 16, Enemy[i].y / 16)) == TRUE) {
            int FQ = 0;
            switch(Enemy[i].typ) {
            case 4: FQ = 4; break;
            case 5: FQ = 5; break;
            case 7: FQ = 3; break;
            }

            if(rand() % FQ < 2) { // INT(RND(1) * FQ)
              Enemy[i].action = 9;
            }
          }
        }
        break;
      }
    }

    if(Enemy[i].typ != 3) {
      Enemy[i].ox = Enemy[i].x / 16;
      Enemy[i].oy = Enemy[i].y / 16;
    }
  }

  // Move the enemy
  switch(Enemy[i].typ) {
  case 1: // Bee
    if(Enemy[i].dir != 4) {
      int s = 0; // 1 ??
      if(Enemy[i].action > 0) s = 2; else s = 1;
      Enemy[i].x += dx[Enemy[i].dir] * s;
      Enemy[i].y += dy[Enemy[i].dir] * s;
    }
    Enemy[i].frame++;
    if(Enemy[i].frame == 3) Enemy[i].frame = 0;
    Enemy[i].z += Enemy[i].az;
    if(Enemy[i].z == 0 || Enemy[i].z == 8) Enemy[i].az = -Enemy[i].az;
    break;
  case 2: // Worm
    if(Enemy[i].dir != 4) {
      Enemy[i].x += dx[Enemy[i].dir];
      Enemy[i].y += dy[Enemy[i].dir];
      Enemy[i].frame += Enemy[i].aframe;
      if(Enemy[i].aframe == 1) {
        if(Enemy[i].frame == 8) Enemy[i].frame = 0;
      } else {
        if(Enemy[i].frame == -1) Enemy[i].frame = 7;
      }
    }
    break;
  case 3: // Shark
    if(Enemy[i].dir != 4) {
      Enemy[i].x += Enemy[i].ox;
      Enemy[i].y += Enemy[i].oy;
    }
    Enemy[i].frame += Enemy[i].aframe;
    if(Enemy[i].frame == 0 || Enemy[i].frame == 8) Enemy[i].aframe = -Enemy[i].aframe;
    break;
  case 4: // Ghost
    if(Enemy[i].dir != 4) {
      if(Enemy[i].action == 0) {
        Enemy[i].x += dx[Enemy[i].dir];
        Enemy[i].y += dy[Enemy[i].dir];
        Enemy[i].z += Enemy[i].az;
        if(Enemy[i].z == 0 || Enemy[i].z == 11) Enemy[i].az = -Enemy[i].az;
      }
    } else {
      Enemy[i].z += Enemy[i].az;
      if(Enemy[i].z == 0 || Enemy[i].z == 11) Enemy[i].az = -Enemy[i].az;
    }
    break;
  case 5: // Putty
    if(Enemy[i].dir != 4) {
      if(Enemy[i].action == 0) {
        Enemy[i].x += dx[Enemy[i].dir] * 2;
        Enemy[i].y += dy[Enemy[i].dir] * 2;
        Enemy[i].frame += Enemy[i].aframe;
        if(Enemy[i].frame == 0 || Enemy[i].frame == 5)
          Enemy[i].aframe = -Enemy[i].aframe;
      }
    } else {
      Enemy[i].frame += Enemy[i].aframe;
      if(Enemy[i].frame == 0 || Enemy[i].frame == 5)
        Enemy[i].aframe = -Enemy[i].aframe;
    }
    break;
  case 6: // Mouse
    if(Enemy[i].dir != 4) {
      Enemy[i].x += dx[Enemy[i].dir] * 2;
      Enemy[i].y += dy[Enemy[i].dir] * 2;
      Enemy[i].frame += Enemy[i].aframe;
      if(Enemy[i].frame == 0 || Enemy[i].frame == 5)
        Enemy[i].aframe = -Enemy[i].aframe;
    }
    break;
  case 7: // Penguin
    if(Enemy[i].dir != 4) {
      if(Enemy[i].action == 0) {
        Enemy[i].x += dx[Enemy[i].dir];
        Enemy[i].y += dy[Enemy[i].dir];
        Enemy[i].frame += Enemy[i].aframe;
        if(Enemy[i].frame == 0 || Enemy[i].frame == 5)
          Enemy[i].aframe = -Enemy[i].aframe;
      }
    }
    break;
}
```

```
      // Has the enemy got any player?
      int Collision = Collide(Enemy[i].x, Enemy[i].y);
      if(Collision > -1) {
        Player[Collision].status = 201;
        Player[Collision].dir = -8;
        Player[Collision].speed = 2;
        Player[Collision].dead = TRUE;
        Player[Collision].lives--;
        if(Player[Collision].lives == -1) Player[Collision].levelreached = Game.area * 5 + Game.level;
        PlaySound(16);
      }

    // The enemy is dead and it's bouncing on the screen
    } else if(Enemy[i].typ < 0) {

      if(Enemy[i].x + Enemy[i].ox < 16 || Enemy[i].x + Enemy[i].ox > 288) Enemy[i].ox = -Enemy[i].ox;

      Enemy[i].x += Enemy[i].ox;
      Enemy[i].y += Enemy[i].oy;
      if(Enemy[i].y > 199) Enemy[i].y = -15;

      if(Enemy[i].action == FALSE) {
        if(Enemy[i].oy <= abs(Enemy[i].ox) * 2) Enemy[i].oy++;
        if(Enemy[i].oy > abs(Enemy[i].ox) * 2 && Enemy[i].x % 16 == 0) {
          Enemy[i].ox = 0;
          Enemy[i].action = TRUE;
        }
      Enemy[i].frame = (Enemy[i].frame + 1) % 4;
      } else {
        if(Enemy[i].oy > 1) Enemy[i].oy--;

        if(Enemy[i].y > 0 && Enemy[i].y < 176) {
          if(Enemy[i].y % 16 == 0) {
            int Fall;
            if(Enemy[i].z > 0) Enemy[i].z--;
            if(Enemy[i].z == 0) {
              Fall = TRUE;
            } else {
              if(rand() % 3 == 0) Fall = TRUE; else Fall = FALSE; //INT(RND(1) * 3)
            }

            if(cell[Enemy[i].y / 16][Enemy[i].x / 16].st == 0 && Fall) {
              Enemy[i].typ = 0;

              int ii = GetFreeObject();
              Object[ii].typ = 18 + rand() % 8; //INT(RND(1) * 8)
              Object[ii].x = Enemy[i].x / 16;
              Object[ii].y = Enemy[i].y / 16;
              Object[ii].time = 100;
              Game.objects++;
            }
          }
        }

        Enemy[i].frame = (Enemy[i].frame + 1) % 8;
      }
    }
  }
}

// Get player input from selected control method and sets the right action
void GetAction(int PlayerNum)
{
  if(Player[PlayerNum].dead != FALSE) return;

  if(Game.mode == NORMAL || Game.mode == POTIONBONUS) {
    // Process player input
    // TEMPORARY CODE
    if(keys[SDLK_LCTRL]) { // Fire button pressed
      Player[PlayerNum].action = 2;
    } else if(keys[SDLK_DOWN]) { // Going down
      Player[PlayerNum].dir = 0;
      if(cell[Player[PlayerNum].y / 16 + 1][Player[PlayerNum].x / 16].st == 0)
        Player[PlayerNum].action = 1;
    } else if(keys[SDLK_LEFT]) { // Going left
      Player[PlayerNum].dir = 1;
      if(cell[Player[PlayerNum].y / 16][Player[PlayerNum].x / 16 - 1].st == 0)
        Player[PlayerNum].action = 1;
    } else if(keys[SDLK_UP]) { // Going up
      Player[PlayerNum].dir = 2;
      if(cell[Player[PlayerNum].y / 16 - 1][Player[PlayerNum].x / 16].st == 0)
        Player[PlayerNum].action = 1;
    } else if(keys[SDLK_RIGHT]) { // Going right
      Player[PlayerNum].dir = 3;
      if(cell[Player[PlayerNum].y / 16][Player[PlayerNum].x / 16 + 1].st == 0)
        Player[PlayerNum].action = 1;
    } else {
      Player[PlayerNum].frame = 2;
    }
  } else {
    // The computer moves the player in demo mode
    int st, FindDir, dd, pd[4];

    srand(time(NULL));

    // Get info on surrounding blocks
    for(int d = 0; d < 4; d++) {
      pd[d] = cell[Player[PlayerNum].y / 16 + dy[d]][Player[PlayerNum].x / 16 + dx[d]].st;
    }
    Player[PlayerNum].action = 0;

    // Get info on the cell the crab is moving onto
    st = cell[Player[PlayerNum].y / 16 + dy[Player[PlayerNum].dir]][Player[PlayerNum].x / 16 + dx[Player
[PlayerNum].dir]].st;
    if(st != 0) {
      // Way blocked!
      FindDir = FALSE;
      if(st == 2) {
        // If the block can be moved, sometimes it's pushed!
        if(rand() % 3 == 0) Player[PlayerNum].action = 2; else FindDir = TRUE;
      }
      if(st == 1 || FindDir == TRUE) {
        // try to find a way out
        for(int d = 0; d < 16; d++) {
          dd = rand() % 4;
          if(pd[dd] == 0) {
            Player[PlayerNum].dir = dd;
            Player[PlayerNum].action = 1;
            break;
          }
        }
      }
    } else {
      if(rand() % 4 == 0) {
        // Moves the crab
        for(int d = 0; d < 16; d++) {
          dd = rand() % 4;
          if(pd[dd] == 0) {
            Player[PlayerNum].dir = dd;
            Player[PlayerNum].action = 1;
            break;
          } else if(pd[dd] == 2) {
            Player[PlayerNum].dir = dd;
            Player[PlayerNum].action = 2;
            break;
          }
        }
      } else {
        Player[PlayerNum].action = 1;
      }
    }
  }
}

// Check players input, moves characters and checks for actions
void MovePlayers()
{
  for(int i = 0; i < Game.players; i++) {
    if(Player[i].status <= 0) {
      // The player can act
      if(Player[i].action == 0)
        GetAction(i);

      // The player is moving
      if(Player[i].action == 1) {
        Player[i].x += dx[Player[i].dir] * Player[i].speed;
        Player[i].y += dy[Player[i].dir] * Player[i].speed;
        Player[i].frame += Player[i].aframe;
        if(Player[i].frame == 0 || Player[i].frame == 5)
          Player[i].aframe = -Player[i].aframe;
        if(Player[i].x % 16 == 0 && Player[i].y % 16 == 0)
          Player[i].action = 0;
      // The player is pushing a block
      } else if(Player[i].action > 1) {
        Player[i].action++;
        if(Player[i].action == 12) Player[i].action = 0;

        int xc = Player[i].x / 16 + dx[Player[i].dir];
        int yc = Player[i].y / 16 + dy[Player[i].dir];
        int xc2 = Player[i].x / 16 + dx[Player[i].dir] * 2;
        int yc2 = Player[i].y / 16 + dy[Player[i].dir] * 2;

        // Push!
        if(Player[i].action == 6) {
          int nnd = cell[yc][xc].nd; // bg
          int nrd = cell[yc][xc].rd; // bonus

          // The block can be moved...
          if(cell[yc][xc].st == 2) {
            // ...but next to it there's another block; the first one is destroyed
            if(cell[yc2][xc2].st > 0) {
              PlaySound(18);

              // Update the background screen buffer
              PutShape(236 + nnd, xc * 16, yc * 16);
              cell[yc][xc].st = 0;
              cell[yc][xc].rd = 0;
              if(cell[yc-1][xc].st > 0)
                PutShape(232 + cell[yc-1][xc].st, xc * 16, yc * 16);
              if(cell[yc+1][xc].st == 0)
                PutShape(236 + cell[yc+1][xc].nd, xc * 16, yc * 16 + 16);

              // Eventually release a bonus
              if(nrd > 0 && Game.status < 1) {
                int ii = GetFreeObject();

                Object[ii].x = xc;
                Object[ii].y = yc;
                Object[ii].typ = nrd;
                if(rand() % 5 == 0) Object[ii].typ = 0; // INT(RND(1) * 5)
                if(Game.mode == DEMO && Object[ii].typ == 14)
                  Object[ii].typ = 0;
                Object[ii].time = 420;
                if(Object[ii].typ > 0)
                  Game.objects++;
              }

              // Add an explosion object
              int ii = GetFreeObject();

              Object[ii].x = xc;
              Object[ii].y = yc;
              Object[ii].typ = 99;
              Object[ii].time = 0;
              if(Game.mode == NORMAL)
                Player[i].score += 10;
                CheckScore(i);

            } else {
              // Initialize the block and add it to the moving block queue
              for(int ii = 0; ii < MAXBLOCKS; ii++)
                if(Block[ii].x == -1) {
                  Block[ii].x = xc;
                  Block[ii].y = yc;
                  Block[ii].ax = dx[Player[i].dir];
                  Block[ii].ay = dy[Player[i].dir];
                  Block[ii].by = i;
                  Block[ii].hitscore = 200;
                  Block[ii].bonus = nrd;

                  // Update the screen
                  PutShape(236 + nnd, Block[ii].x * 16, Block[ii].y * 16);
                  int st = cell[Block[ii].y - 1][Block[ii].x].st;
                  if(st > 0)
                    PutShape(232 + st, Block[ii].x * 16, Block[ii].y * 16);
                  st = cell[Block[ii].y + 1][Block[ii].x].st;
                  int nd = cell[Block[ii].y + 1][Block[ii].x].nd;
                  if(st == 0)
                    PutShape(236 + nd, Block[ii].x * 16, Block[ii].y * 16 + 16);
                  PlaySound(15);
                  break;
                }
            }
          }
        }
      }

      // The player is invulnerable until his status reaches 0
      if(Player[i].status < 0) Player[i].status++;
    } else {
      // Other stuff
      if(Player[i].status >= 1 && Player[i].status <= 200) {
        // The player is trapped; he's free once his status reaches 200
        Player[i].status++;
        if(Player[i].status == 200) Player[i].status = 0;
      } else if(Player[i].status == 201) {
        // The player has been killed
        if(Player[i].y < 200) Player[i].y += Player[i].dir;
        if(Player[i].dir < 10) Player[i].dir++;
      }
    }
  }
}

// Returns -1 if the object at xPos,yPos doesn't collide with a player,
// otherwise returns player's number (0 or 1)
int Collide(int xPos, int yPos)
{
  for(int c = 0; c < Game.players; c++) {
    if(Player[c].dead == FALSE) {
      if(Player[c].status > -1 && Player[c].status < 201) {
        if(xPos > Player[c].x - 12 && xPos < Player[c].x + 12) {
          if(yPos > Player[c].y - 12 && yPos < Player[c].y + 12) {
            return c;
          }
        }
      }
    }
  }
  return -1;
}

// Check if a block is in motion and move it
void MoveBlocks()
{
  int range = 0;

  for(int i = 0; i < MAXBLOCKS; i++) {
    // The block is moving
    if(Block[i].x > -1) {
      // Find the content of the cell the block is moving onto
      int nst, nnd, nrd;

      nst = cell[Block[i].y + Block[i].ay][Block[i].x + Block[i].ax].st;
      nnd = cell[Block[i].y + Block[i].ay][Block[i].x + Block[i].ax].nd;
      nrd = cell[Block[i].y + Block[i].ay][Block[i].x + Block[i].ax].rd;

      // Check if the block strikes an enemy
      for(int E = 0; E < 2; E++) {
        for(int ii = 0; ii < MAXENEMIES; ii++) {
          if(Enemy[ii].typ > 0) {
            if(Enemy[ii].typ == 3) range = 18; else range = 16;
            if(E == 1) range = 15;
            if(((Block[i].x + (Block[i].ax * E)) * 16) > Enemy[ii].x + dx[Enemy[ii].dir] - range &&
               ((Block[i].x + (Block[i].ax * E)) * 16) < Enemy[ii].x + dx[Enemy[ii].dir] + range) {
              if(((Block[i].y + (Block[i].ay * E)) * 16) > Enemy[ii].y + dy[Enemy[ii].dir] - range &&
                 ((Block[i].y + (Block[i].ay * E)) * 16) < Enemy[ii].y + dy[Enemy[ii].dir] + range) {

                // An enemy is gone!
                if(Game.mode == NORMAL) Player[Block[i].by].score += Block[i].hitscore;
                CheckScore(Block[i].by);

                for(int iii = 0; iii < MAXOBJS; iii++)
                  if(Object[iii].typ == 0) {
                    // Creates a score object and erases the enemy
                    Object[iii].typ = Block[i].hitscore + Block[i].by * 10000;
                    Object[iii].time = 0;
                    Object[iii].x = Block[i].x;
                    Object[iii].y = Block[i].y;
                    break;
                  }

                KillEnemy(ii);
                Block[i].hitscore *= 2;
                if(Block[i].hitscore > 6400) Block[i].hitscore = 6400;
              }
            }
          }
        }

        // Erase each shot that hits the block
        for(int ii = 0; ii < MAXSHOTS; ii++) {
          if(Shot[ii].typ > 0 && Shot[ii].typ < 4) {
            if(((Block[i].x + (Block[i].ax * E)) * 16) > Shot[ii].x - 12 &&
               ((Block[i].x + (Block[i].ax * E)) * 16) < Shot[ii].x + 12) {
              if(((Block[i].y + (Block[i].ay * E)) * 16) > Shot[ii].y - 12 &&
                 ((Block[i].y + (Block[i].ay * E)) * 16) < Shot[ii].y + 12) {
                switch(Shot[ii].typ) {
                case 1: Shot[ii].typ = 4; Shot[ii].time = 0; break;
                case 2: Shot[ii].typ = 0; break;
                case 3: Shot[ii].typ = 5; Shot[ii].time = 0; PlaySound(12); break;
                }
              }
            }
          }
        }

        // Finds if a player is being hit by the block
        int Collision = Collide((Block[i].x + (Block[i].ax * E)) * 16, (Block[i].y + (Block[i].ay * E)) *
16);
        if(Collision != -1) {
          // The player is hit and dies
          Player[Collision].status = 201;
          Player[Collision].dir = -8;
          Player[Collision].speed = 2;
          Player[Collision].dead = TRUE;
          Player[Collision].lives--;
          if(Player[Collision].lives == -1) Player[Collision].levelreached = Game.area * 5 + Game.level;
          PlaySound(16);
        }
      }

      if(nst == 0) {
        // The cell the block is moving onto is empty; the block can move
        cell[Block[i].y][Block[i].x].st = 0;
        cell[Block[i].y][Block[i].x].rd = 0;
        Block[i].x += Block[i].ax;
        Block[i].y += Block[i].ay;
        cell[Block[i].y][Block[i].x].st = 2;
        cell[Block[i].y][Block[i].x].nd = nnd;
        cell[Block[i].y][Block[i].x].rd = Block[i].bonus;
      } else {
        // The block can't move; it stops its run.
        PutShape(236, Block[i].x * 16, Block[i].y * 16);

        if(nst == 0)
          PutShape(234, Block[i].x * 16, Block[i].y * 16 + 16);

        // If the block stops over an object, the object is erased
        for(int ii = 0; ii < MAXOBJS; ii++) {
          if(Object[ii].typ > 0)
            if(Object[ii].x == Block[i].x && Object[ii].y == Block[i].y)
              Object[ii].typ = 0;
        }
        // The current block is deleted from the moving block queue
        Block[i].x = -1;
      }
    }
  }
}

// Update Buffer and copy it to the screen
void DrawScreen() // better name: DrawObjects
{
  int F = 0, zd = 0;

  // Draw the shots
  for(int i = 0; i < MAXSHOTS; i++) {
    if(Shot[i].typ > 0) {
      switch(Shot[i].typ) {
      case 1:
        switch(Shot[i].time) {
        case 0 ... 1: F = 164; break;
        case 2: F = 165; break;
        case 3: F = 166; break;
        case 4: F = 167; break;
        default:
          F = 168 + (Shot[i].time / 3) % 3;
        }
        break;
      case 2: F = 176 + Shot[i].time % 4; break;
      case 3: F = 180 + Shot[i].time % 4; break;
      case 4: F = 173 + Shot[i].time / 2; break;
      case 5: F = 184 + Shot[i].time / 2; break;
      }

      PutShape(F, Shot[i].x, Shot[i].y);
    }
  }

  // Objects
  for(int i = 0; i < MAXOBJS; i++) {
    if(Object[i].typ > 0) {
      switch(Object[i].typ) {
      case 1 ... 33:
        PutShape(189 + Object[i].typ, Object[i].x * 16, Object[i].y * 16);
        break;
      case 97:
        PutShape(223 + Object[i].time / 4, Object[i].x * 16, Object[i].y * 16);
        break;
      case 98:
        PutShape(187 + Object[i].time / 4, Object[i].x * 16, Object[i].y * 16);
        break;
      case 99:
        PutShape(230 + Object[i].time / 3, Object[i].x * 16, Object[i].y * 16);
        break;
      default:
        if(Object[i].typ > 99) {
          int c = 0, Points = 0;

          if(Object[i].typ > 10000) {
            Points = Object[i].typ - 10000;
            c = 44;
          } else {
            Points = Object[i].typ;
            c = 12;
          }
          if(Object[i].time < 7) {
            PutScore(Points, Object[i].x * 16, Object[i].y * 16 - Object[i].time, c);
          } else {
            PutScore(Points, Object[i].x * 16, Object[i].y * 16 - 6, c);
          }
        }
      }
    }
  }

  // Draws the lightnings over each enemy whenever the player has taken the bonus
  if(Game.status == -2) {
    srand(time(NULL));
    for(int i = 0; i < MAXENEMIES; i++) {
      if(Enemy[i].typ > 0) {
        for(int ii = 0; ii < 3; ii++) {
          int Oldx, Newx;
          Oldx = rand() % 8; //INT(RND(1) * 8)
          for(int iii = 0; iii <= Enemy[i].y - 2; iii += 12 - ii * 2) {
            Newx = rand() % 8; //INT(RND(1) * 8)
            BlastLine(Enemy[i].x + 4 + Oldx, iii, Enemy[i].x + 4 + Newx, (iii + (16 - (ii * 4))), (48 -
(ii * 2)));
            Oldx = Newx;
          }
        }
      }
    }
  }

  // Draw enemies
  for(int i = 0; i < MAXENEMIES; i++) {
    if(Enemy[i].typ > 0) {
      switch(Enemy[i].typ) {
      case 1:
        zd = 3;
        F = 40 + Enemy[i].dir * 3 + Enemy[i].frame;
        if(Enemy[i].dir == 4) F = 40 + Enemy[i].frame;
        break;
      case 2:
        zd = 1;
        if(Enemy[i].dir == 0 || Enemy[i].dir == 2) F = 52 + Enemy[i].frame; else F = 60 + Enemy[i].frame;
      if(Enemy[i].dir == 4) F = 60;
      break;
    case 3:
      zd = 1;
      F = 68 + Enemy[i].dir * 3 + Enemy[i].frame / 3;
      if(Enemy[i].dir == 4) F = 68 + Enemy[i].frame / 3;
      break;
    case 4:
      zd = 4;
      F = 80 + Enemy[i].dir * 3;
      if(Enemy[i].action > 0) F = 80 + Enemy[i].dir * 3 + Enemy[i].action / 4;
      if(Enemy[i].dir == 4) F = 80;
      break;
    case 5:
      zd = 1;
      F = 92 + Enemy[i].dir * 3 + Enemy[i].frame / 2;
      if(Enemy[i].action > 0) F = 104 + Enemy[i].dir;
      if(Enemy[i].dir == 4) F = 92 + Enemy[i].frame / 2;
      break;
    case 6:
      zd = 1;
      F = 108 + Enemy[i].dir * 3 + Enemy[i].frame / 2;
      if(Enemy[i].dir == 4) F = 109;
      break;
    case 7:
      zd = 1;
      F = 120 + Enemy[i].dir * 3 + Enemy[i].frame / 2;
      if(Enemy[i].action > 0) F = 132 + Enemy[i].dir;
      if(Enemy[i].dir == 4) F = 121;
    }

    if(Game.status == -2) {
      PutShape(F, Enemy[i].x, Enemy[i].y - rand() % 2);
    } else {
      PutShape(F, Enemy[i].x, Enemy[i].y + Enemy[i].z / zd - 1);
    }
  } else if(Enemy[i].typ < 0) {
    F = 135 + abs(Enemy[i].typ) + ((Enemy[i].frame / ((2 + (Enemy[i].action == FALSE)))) * 7);
    PutShape(F, Enemy[i].x, Enemy[i].y);
  }
}

// Draws moving blocks only
for(int i = 0; i < MAXBLOCKS; i++) {
  if(Block[i].x > -1) {
    PutShape(236, Block[i].x * 16, Block[i].y * 16);
    if(cell[Block[i].y + 1][Block[i].x].st == 0)
      PutShape(234, Block[i].x * 16, Block[i].y * 16 + 16);
  }
}

// And finally draw the players
for(int i = 0; i < Game.players; i++) {
  if(Player[i].dead == FALSE || Player[i].dead == 2) {
    switch(Player[i].action) {
    case 0:
    case 1:
      F = i * 20 + Player[i].dir * 4 + Player[i].frame / 2;
      break;
    case 3:
    case 11:
      F = i * 20 + Player[i].dir * 4 + 1;
      break;
    default:
      F = i * 20 + Player[i].dir * 4 + 3;
    }

    if(Player[i].status != 0) {
      switch(Player[i].status) {
      case 1 ... 189:
        F = 172 - i;
        PutShape(F, Player[i].x, Player[i].y + (Player[i].status / 8) % 2 - 1);
        break;
      case 190 ... 195:
        PutShape(i * 20 + 18, Player[i].x, Player[i].y);
        break;
      case 196 ... 197:
        PutShape(i * 20 + 17, Player[i].x, Player[i].y);
        break;
      case 198 ... 199:
        PutShape(i * 20 + 16, Player[i].x, Player[i].y);
        break;
      case 201:
        PutShape(i * 20 + 19, Player[i].x, Player[i].y);
        break;
      default:
        if(Player[i].status < 0)
          if((abs(Player[i].status) / 3) % 2 == 0)
            PutShape(F, Player[i].x, Player[i].y);
        break;
      }
    } else {
      if(Game.status < 501) {
        PutShape(F, Player[i].x, Player[i].y);
      } else {
        PutShape(i * 20 + 16 + (Game.status - 500) / 8, Player[i].x, Player[i].y);
      }
    }
  }
}

// If the time has passed, draws the flame-thingies
if(Game.time <= 0) {
  if(Game.status > -501 && Game.status < 1) {
    for(int i = 0; i < Game.players; i++) {
      if(Player[i].dead == FALSE)
        PutShape((226 + Death[i].frame), (int)Death[i].x, (int)Death[i].y);
    }
  }
}

SDL_Rect dst;
dst.x = 0;
dst.y = 192;
dst.w = 320;
dst.h = 8;
SDL_FillRect(gamescreen, &dst, 0);

//Prints the players stuff on the bottom of the screen
if(Game.mode == DEMO) {
  SPrint("rr DEMO MODE ss", 100, 192, 56);
} else {
  char string[64];

  if(Game.status == -501) {
    sprintf(string, "COLLECT ALL ITEMS! - TIME: %i", Game.time);
    SPrint(string, 28, 192, 56);
  } else {
    for(int i = 0; i < Game.players; i++) {
      if(Player[i].lives > 9) {
        strcpy(string, "e9");
      } else if(Player[i].lives > -1) {
        sprintf(string, "e%i", Player[i].lives);
      } else {
        strcpy(string, "e-");
      }
      strcat(string, " ");

      char str[64];
      if(Player[i].score > 9999999) Player[i].score = 9999999;
```

```
        sprintf(str,"%07i ", Player[i].score);
        strcat(string, str);
        strcat(string, Player[i].bonus);
        SPrint(string, i * 192, 192, i * 32 + 8);
      }

      //sprintf(string, "fg %i-%i ", Game.area + 1, Game.level + 1);
      //SPrint(string, ((320 - (strlen(string) * 8) + 8) / 2), 192, 56);

      sprintf(string, "TIME: %i", (Game.time > 0 ? Game.time : 0));
      SPrint(string, 128, 192, 56);
    }
  }
}

void RedrawLevel()
{
  SDL_FillRect(gamescreen, NULL, 0);

  for(int y = 11; y >= 0; y--)
    for(int x = 0; x < 20; x++) {
      PutShape((236 + cell[y][x].nd), x * 16, y * 16); // bg

      if(cell[y][x].st > 0) {
        PutShape(234 + cell[y][x].st, x * 16, y * 16); // fg
        if(y < 11 && cell[y+1][x].st == 0) {
            PutShape(232 + cell[y][x].st, x * 16, (y + 1) * 16); // shade FIX LATER
        }
      }
    }
}

// Display an animated message
void ShowAreaIntro()
{
  char WTitle[64];
  char ATitle[64];
  char PassWord[6] = "?????";
  char str[64];

  strcpy(WTitle, wwd->title);
  strcpy(ATitle, (wwd->area + Game.area)->title);
  memcpy(PassWord, wwd->pass[Game.area], 4);

  // If in demo mode the password isn't shown
  if(Game.mode == DEMO)
    strcpy(PassWord, "?????");
  else {
    PassWord[4] = Game.players | 0x30;
    PassWord[5] = 0;
  }

  // Clear the buffer and print area informations on it
  SDL_FillRect(gamescreen, 0, 0);
  sprintf(str, "ENTERING AREA %i ...", Game.area + 1);
  SPrint(str, ((320 - (strlen(str) * 8)) / 2), 60, 56);

  // Draw the living crabs at the center of the screen
  if(Game.players == 1) {
    PutShape(1, 152, 110);
  } else {
    if(Player[0].lives == -1) {
      PutShape(21, 152, 110);
    } else if(Player[1].lives == -1) {
      PutShape(1, 152, 110);
    } else {
      PutShape(1, 132, 110);
      PutShape(21, 172, 110);
    }
  }
  SPrint(WTitle, ((320 - (strlen(WTitle) * 8)) / 2), 50, 56);
  SPrint(PassWord, 140, 140, 56);

  float angle = 0;
  for(int j = 0; j < 200; j++) {
    SDL_Rect dst;

    dst.x = 0;
    dst.y = 80;
    dst.w = 320;
    dst.h = 18;
    SDL_FillRect(gamescreen, &dst, 0);

    // Draw the waving area title
    for(int i = 0; i < (int)strlen(ATitle); i++) {
      char c[2];
      c[0] = ATitle[i]; c[1] = 0;
      SPrint(c, (((320 - (strlen(ATitle) * 8)) / 2) + ((i - 1) * 8)), (86 + (sin(angle + (float)i / 2) *
6)), 192);
    }
    angle += .3;
    if(angle > 6.28) angle = 0;

    if(SDL_Pressed()) break;
    BlitAndWait(2);
  }

}

// Load new area data from the world data file, displaying an animated message
void InitArea(int a) // NewArea
{
  // Move tiles from area to sprites
  SDL_SetPalette(sprites, SDL_LOGPAL, (wwd->area + a)->pal, 240, 16);
  SDL_SetPalette(gamescreen, SDL_LOGPAL, (wwd->area + a)->pal, 240, 16);
  for(int i = 0; i < 5; i++) {
    SDL_Rect src, dst;

    src.x = 0;
    src.y = i * 16;
    src.w = 16;
    src.h = 16;

    dst.x = 240 + i * 16;
    dst.y = 176;

    SDL_BlitSurface((wwd->area + a)->sprites, &src, sprites, &dst);
  }

  // init text color for each area
  Game.textcol = (wwd->area + Game.area)->textcol;

  // load midi from _full_ path
  char midi[256];
  sprintf(midi, "%s/%s", wwd->fullpath, (wwd->area + Game.area)->midifile);

  LoadMIDI(midi);
}

// Load a level from world data file
void InitLevel(int a, int l) // LoadLevel
{
  Game.monsters = 0;

  // Load the level shape
  memcpy(cell, (wwd->area + a)->level[l].cell, sizeof(cell));

  // Get level time
  Game.time = (wwd->area + a)->level[l].time;
```

```c
  // Get enemies
  memcpy(Enemy, (wwd->area + a)->level[l].enemy, sizeof(Enemy));
  for(int i = 0; i < MAXENEMIES; i++) {
    if(Enemy[i].typ > 0) Game.monsters++;
  }

  // Get players starting positions
  Player[0].x = (wwd->area + a)->level[l].x1;
  Player[0].y = (wwd->area + a)->level[l].y1;
  Player[1].x = (wwd->area + a)->level[l].x2;
  Player[1].y = (wwd->area + a)->level[l].y2;

  // Clear screen buffer and draws the new level on it
  RedrawLevel();

  // Initialize game variables
  memset(Block, 0, sizeof(Block));
  for(int i = 0; i < MAXBLOCKS; i++) Block[i].x = -1;
  memset(Object, 0, sizeof(Object));
  memset(Shot, 0, sizeof(Shot));

  Game.objects = 0;
  Game.status = 0;
  Game.special = FALSE;
  if(Game.mode == POTIONBONUS) Game.mode = NORMAL;

  // Initialize enemies variables
  for(int i = 0; i < MAXENEMIES; i++) {
    if(Enemy[i].typ > 0 && Enemy[i].typ != 3) {
      Enemy[i].dir = 4;
      Enemy[i].ox = Enemy[i].x / 16;
      Enemy[i].oy = Enemy[i].y / 16;
    }

    if(Enemy[i].typ == 1) {
      Enemy[i].z = rand() % 7; //INT(RND(1) * 7)
      Enemy[i].az = 1;
    } else if(Enemy[i].typ == 4) {
      Enemy[i].z = rand() % 10; //INT(RND(1) * 10)
      Enemy[i].az = 1;
    } else {
      Enemy[i].z = 1;
      Enemy[i].az = 0;
    }
  }

  // Also initializes players at their starting positions
  Player[0].dead = TRUE;
  Player[1].dead = TRUE;

  for(int i = 0; i < Game.players; i++) {
    Player[i].dead = FALSE;
    if(Player[i].lives == -1) Player[i].dead = 3;
    Player[i].status = -120;
    Player[i].dir = 0;
    if(Player[i].speed == 1) Player[i].speed = 2;
    Player[i].frame = 2;
    Player[i].action = 0;
    Player[i].potion = 0;
  }

  Blocked = FALSE;
}

void ReadyToStart()
{
  char line[64];

  sprintf(line, "AREA %i-%i", Game.area + 1, Game.level + 1);
```

```c
    for(int o = 0; o < 4; o++) {
      SPrint(line, ((320 - (strlen(line) * 8)) / 2), 88, Game.textcol);
      SPrint("GET READY!", 120, 96, Game.textcol);
      DrawScreen();
      BlitAndWait(24);
      RedrawLevel();
      DrawScreen();
      BlitAndWait(24);
    }
}

int AbortGameYN()
{
    int choice = 1;
    int result = FALSE;

    TimerOn = FALSE;
    PlaySound(14);
    keys[SDLK_ESCAPE] = 0;

    while(1) {
      if(keys[SDLK_LEFT]) choice = 0;
      else if(keys[SDLK_RIGHT]) choice = 1;
      else if(keys[SDLK_ESCAPE]) {
        result = FALSE;
        break;
      } else if(keys[SDLK_LCTRL] || keys[SDLK_RETURN]) {
        result = (choice == 0 ? TRUE : FALSE);
        break;
      }

      RedrawLevel();
      DrawScreen();

      SPrint("ABORT CURRENT GAME (YES/NO)?", 60, 96, Game.textcol);
      if(choice == 0) {
        PutBox(60+20*8-1, 96-1, 60+23*8, 96+8, 83);
      } else {
        PutBox(60+24*8-1, 96-1, 60+26*8, 96+8, 83);
      }

      BlitAndWait(2);
    }

    keys[SDLK_ESCAPE] = 0;
    if(Game.monsters > 0) TimerOn = TRUE;
    return result;
}

void PlayGame()
{
    int AbortFlag = FALSE;
    int LevelPass = TRUE;

    // Set the default game palette
    SDL_SetPalette(gamescreen, SDL_LOGPAL, gamepal, 0, 256);
    SDL_FillRect(gamescreen, NULL, 0);

    // Initialize players variables
    for(int i = 0; i < Game.players; i++) {
      Player[i].frame = 2;
      Player[i].aframe = 1;
      Player[i].score = 0;
      Player[i].nextextra = 30000;
      Player[i].speed = 2;
      Player[i].status = 0;
      Player[i].lives = 3;
      strcpy(Player[i].bonus, "hhhhh");
      Player[i].dir = 0;
    Player[i].levelreached = Game.area * 5 + Game.level;
  }

  Game.status = 0;
  Game.numareas = wwd->numofareas;

  // If we're in demo mode, choose a random level
  if(Game.mode == DEMO) {
    srand(time(NULL));
    Game.area = rand() % 10;
    if(Game.area > Game.numareas) Game.area = Game.numareas - 1;
    Game.level = rand() % 5;
  }

_reinit_area:
  InitArea(Game.area);
  ShowAreaIntro();

  do {
    InitLevel(Game.area, Game.level);

    // Blink "Ready to start"
    ReadyToStart();

    // Starts the game
    AbortFlag = FALSE;
    PlayMIDI();

    TimerOn = TRUE;

    // Begin game loop
    do {
      RedrawLevel();
      MovePlayers();
      MoveBlocks();
      MoveEnemies();
      MoveDeath();
      MoveShots();
      if(Game.status == -501) {
        // We're in potion bonus
        HandlePotion();
      } else {
        // Normal objects handling
        HandleObjects();
      }
      CheckStatus();
      CheckTime();

      // The ESC key is pressed TODO: put yes/no menu
      if(keys[SDLK_ESCAPE]) {
        if(Game.mode == DEMO) {
          AbortFlag = TRUE;
          break;
        } else {
          AbortFlag = AbortGameYN();
          if(AbortFlag == TRUE) break;
        }
      }

      DrawScreen();
      BlitAndWait(2);
    } while(Game.status < 500);

    // The game is halted
    TimerOn = FALSE;
    StopMIDI();

    // If the ESC key has been pressed abort the game
    if(AbortFlag == TRUE) goto _exit;
  // Find if the level has been finished
  LevelPass = TRUE;
  if(Game.players == 1) {
    if(Player[0].lives == -1) break;
    if(Player[0].dead != FALSE) LevelPass = FALSE;
  } else {
    if(Player[0].lives == -1 && Player[1].lives == -1) break;
    if(Player[0].dead != FALSE && Player[1].dead != FALSE) LevelPass = FALSE;
  }

  // The level is finished; do the victory scene
  if(LevelPass == TRUE) {
    for(int i = 0; i < Game.players; i++)
      Player[i].status = 0;

    PlaySound(13);
    do {
      Game.status++;
      RedrawLevel();
      DrawScreen();
      BlitAndWait(2);
    } while(Game.status < 516);

    // Checks if all the blocks have been destroyed
    int SpecialBonus = TRUE;
    for(int i = 0; i < 240; i++) {
      if(cell[i / 20][i % 20].st == 2) {
        SpecialBonus = FALSE;
        break;
      }
    }

    // All the blocks have been destroyed: give the special bonus!
    if(SpecialBonus == TRUE && Game.mode == NORMAL) {
      PlaySound(9);
      for(int i = 0; i < Game.players; i++) {
        if(Player[i].lives > -1) {
          Player[i].score += 50000;
          CheckScore(i);
        }
      }
      DrawScreen();
      SPrint("TOTAL DESTRUCTION BONUS", 68, 92, Game.textcol);
      SPrint("50000 PTS", 124, 101, Game.textcol);
    }

    BlitAndWait(100);

    // Increase the level
    Game.level++;
    if(Game.level > 4) {
      Game.level = 0;
      Game.area++;
      if(Game.area < Game.numareas) {
        goto _reinit_area;
      } else {
        // No more areas available; shows the end of the game!
        for(int i = 0; i < Game.players; i++) {
          if(Player[i].lives > -1)
            Player[i].levelreached = Game.area * 5 + Game.level;
        }
        //TheEnd();
        break;
      }
    }
  }

  // If we're in demo mode, end it
  if(Game.mode == DEMO) break;
} while(1);
```

```
  // Game over
  //CLS
  if(Game.mode == NORMAL) {
    SPrint("GAME OVER", 124, 96, 56);
    //Fade 1
    SDL_PurgeEvents();
    for(int i = 0; i < 150; i++) {
      if(SDL_Pressed()) break;
      BlitAndWait(2);
    }
    //Fade 0
  }
_exit:
  SDL_SetPalette(gamescreen, SDL_LOGPAL, syspal, 0, 256);
}

void InitGame()
{
  // load palettes
  LoadPalette("data/palette0.pal", gamepal, 256);
  LoadPalette("data/palette1.pal", enemypal[0], 80);
  LoadPalette("data/palette2.pal", enemypal[1], 80);
  LoadPalette("data/palette3.pal", enemypal[2], 80);
  LoadPalette("data/palette4.pal", syspal, 256);

  // load font
  LoadFont("data/font.dat");

  // and initialize surfaces
  gamescreen = SDL_CreateRGBSurface(SDL_SWSURFACE, 320, 200, 8, 0, 0, 0, 0);
  SDL_SetPalette(gamescreen, SDL_LOGPAL, gamepal, 0, 256);

  // load sprites and screens
  logo = SDL_LoadBMP("data/logo.bmp");
  title = SDL_LoadBMP("data/title.bmp");
  theend = SDL_LoadBMP("data/theend.bmp");
  sprites = SDL_LoadBMP("data/sprites.bmp");

  SDL_SetColorKey(sprites, SDL_SRCCOLORKEY, *(char *)(sprites->pixels));

  // load sounds
  for(int i = 0; i < MAXSOUNDS; i++) {
    char str[64];
    sprintf(str, "data/snd%i.voc", i);

    sfx[i] = Mix_LoadWAV(str);
  }

  // load default world and its music
  LoadWorld("./world/WETSPOT2.WWD");
}

#undef main
int main()
{
  int result;

  result = SDL_Init(SDL_INIT_EVERYTHING);
  if(result) {
    printf("Failed to init SDL\n");
    exit(1);
  }

  atexit(SDL_Quit);

  screen = SDL_SetVideoMode(320, 240, 16, SDL_SWSURFACE);

  SoundInit();
  TimerInit();
  InitGame();

  keys = SDL_GetKeyState(NULL); // InputInit(); joystick also

  Logo();
  Intro();
  Menu(); // PlayGame is called from menu

  TimerDeinit();
  return 0;
}
```