Bomb Rain

An unexplainable flurry of explosives are falling from the sky. There's no time to wonder where these deadly projectiles are coming from when you've got to dance to avoid the BOMB RAIN!

The most exciting part of this program can't be conveyed by pictures alone because this game has a sound track. Utilizing allegro's sound generation functions Bomb Rain comes with a sound track to accompany your fight for your life. Using this code it should be possible to add sound to any game.

Bomb Rain is by Simon Parzer, written for the MinorHack challenge on October 14, 2006.

// bombrain.c listing begins:

```
#include "allegro.h"
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <string>
#include <vector>
#include <ctime>

using namespace std;

#define FRAME_RATE 60
#define MUSIC_RATE 8

BITMAP* buffer;
volatile int sC = 0;

typedef struct
{
  SAMPLE* square1;
  SAMPLE* square2;
  SAMPLE* saw;
  SAMPLE* noise;
} Samples_t;

Samples_t Samples;

void init_samples( void )
{
  int sq1_len = 100;
  int sq2_len = 200;
  int saw_len = 400;
  int noise_len = 5000;
  Samples.square1 = create_sample(8,0,22050,sq1_len);
  unsigned char* square1 = (unsigned char*)Samples.square1->data;
  Samples.square2 = create_sample(8,0,22050,sq2_len);
  unsigned char* square2 = (unsigned char*)Samples.square2->data;
  Samples.saw = create_sample(8,0,22050,saw_len);
  unsigned char* saw = (unsigned char*)Samples.saw->data;

  for (int i=0; i<sq1_len; ++i)
  {
    if (i<sq1_len/3)
      square1[i] = 200;
    else
      square1[i] = 0;
  }
  for (int i=0; i<sq2_len; ++i)
  {
    if (i<sq2_len/2)
      square2[i] = 200;
    else
      square2[i] = 0;
  }
  for (int i=0; i<saw_len; ++i)
  {
    saw[i] = 255 - (int)((double)i/(double)saw_len*(double)255);
  }

  Samples.noise = create_sample(8,0,22050,noise_len);
  unsigned char* noise = (unsigned char*)Samples.noise->data;
```

// Listing continued on next page...

// Listing continued from previous page

```
  for (int i=0; i<noise_len; ++i)
  {
    noise[i] = rand()%255;
  }
}

void destroy_samples( void )
{
  destroy_sample(Samples.square1);
  destroy_sample(Samples.square2);
  destroy_sample(Samples.saw);
  destroy_sample(Samples.noise);
}

typedef struct
{
  int pos;
  string square1, square2, saw, noise;
} Music_t;
Music_t Music;

void clear_music( void )
{
  Music.square1.clear();
  Music.square2.clear();
  Music.saw.clear();
  Music.noise.clear();
  Music.pos = 0;
}

void add_music(const string& square1, const string& square2,
  const string& saw, const string& noise)
{
  if (square1.size() != square2.size() || square1.size() != saw.size()
     || square1.size() != noise.size())
  {
    return; //Error adding music
  }
  Music.square1.append(square1);
  Music.square2.append(square2);
  Music.saw.append(saw);
  Music.noise.append(noise);
}

void play_music_unit( SAMPLE* s, char note )
{
  static int pitch_vals[] =
    {1000,1059,1122,1189,1260,1335,1414,1498,1587,1681,1781,1887};
  static char pitch_notes[] = "cCdDefFgGaAb";
  if (note == '.')
  {
    stop_sample(s);
  }
  for (int i=0; i<12; ++i)
  {
    if (note == pitch_notes[i])
    {
      stop_sample(s);
      play_sample(s,255,127,pitch_vals[i],1);
      break;
    }
  }
}
```

// Listing continued on next page...

// Listing continued from previous page

```
void update_music( void )
{
  static int count = 0;
  ++count;
  if (count >= FRAME_RATE/MUSIC_RATE)
  {
    count = 0;
    int p = Music.pos;
    play_music_unit(Samples.square1,Music.square1[p]);
    play_music_unit(Samples.square2,Music.square2[p]);
    play_music_unit(Samples.saw,Music.saw[p]);
    play_music_unit(Samples.noise,Music.noise[p]);
    Music.pos++;
    if (Music.pos >= Music.square1.size())
    {
      Music.pos = 0;
    }
  }
}

void sC_int( void )
{
  ++sC;
}

void swap_buffers( void )
{
  blit(buffer,screen,0,0,0,0,SCREEN_W,SCREEN_H);
}

const int BOTTOM = 420;

typedef struct
{
  int x, y;
  int explode_state; // 0 = flying, >0 = exploding
  int movement_state; // How many pixels moved
} Bomb;

vector<Bomb> bombs;

void add_random_bomb( void )
{
  Bomb b;
  b.x = rand()%(SCREEN_W-10)+5;
  b.y = -50;
  b.explode_state = 0;
  bombs.push_back(b);
}

void draw_bombs( void )
{
  long color = 0xcccccc;
  for (int i=0; i<bombs.size(); ++i)
  {
    if (!bombs[i].explode_state)
    {
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x-5,bombs[i].y-4,color);
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x+5,bombs[i].y-4,color);

      line(buffer,bombs[i].x-5,bombs[i].y-4,bombs[i].x-5,bombs[i].y-10,color);
      line(buffer,bombs[i].x+5,bombs[i].y-4,bombs[i].x+5,bombs[i].y-10,color);
```

// Listing continued on next page...

// Listing continued from previous page

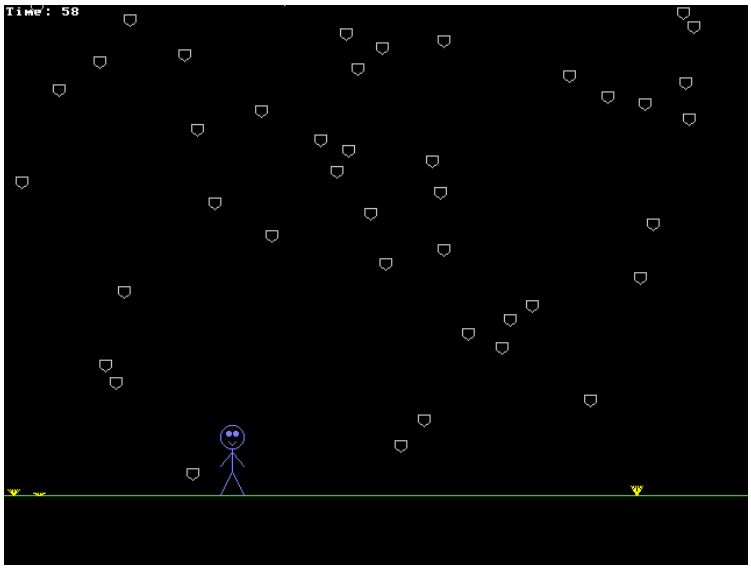
```
      line(buffer,bombs[i].x-5,bombs[i].y-10,bombs[i].x+5,bombs[i].y-10,color);
    }
    else if (bombs[i].explode_state<=10)
    {
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x-5,
        bombs[i].y-bombs[i].explode_state,0xffff00);
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x-3,
        bombs[i].y-bombs[i].explode_state,0xffff00);
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x-1,
        bombs[i].y-bombs[i].explode_state,0xffff00);
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x+1,
        bombs[i].y-bombs[i].explode_state,0xffff00);
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x+3,
        bombs[i].y-bombs[i].explode_state,0xffff00);
      line(buffer,bombs[i].x,bombs[i].y,bombs[i].x+5,
        bombs[i].y-bombs[i].explode_state,0xffff00);
    }
  }
}

int player_height = 60;
int player_x = 100;

bool game_over = false;

void advance_bombs( void )
{
  for (int i=0; i<bombs.size(); ++i)
  {
    Bomb& b = bombs[i];
    if (!b.explode_state)
    {
      ++b.y;
      ++b.y;
      if (b.y>100) ++b.y;
      if (b.y>200) ++b.y;
      if (b.y>300) ++b.y;
      if (b.y>350) ++b.y;
      if (b.y>370) { ++b.y; ++b.y; }

      if (b.y >= BOTTOM-player_height
        && (b.x >= player_x-10 && b.x <=player_x+10))
      {
        b.y = BOTTOM - player_height;
        b.explode_state = 1;
        game_over = true;
        static bool start_music = true;
        if (start_music) {
          clear_music();
          //---------|---|---|---|---|---|---|---|---
          add_music("d   c d e   f e ",
                    "f   e f g   a g ",
                    "a      .C      .",
                    "                ");
          //---------|---|---|---|---|---|---|---|---
          add_music("d   c d e   f e ",
                    "f   e f g   a g ",
                    "a      .C      .",
                    "                ");
          //---------|---|---|---|---|---|---|---|---
          add_music("c   d                          ",
                    "D   f                          ",
```



// Listing continued on next page...

// Listing continued from previous page

```
                    "g   a                              ",
                    "                                   ");
          //---------|---|---|---|---|---|---|---|---
          add_music(".                                   ",
                    ".                                  ",
                    ".                                  ",
                    ".                                  ");
          start_music = false;
        }
      }
      if (b.y >= BOTTOM)
      {
        b.y=BOTTOM;
        b.explode_state = 1;
      }
    }
    else
    {
      if (b.explode_state <= 10)
      {
        ++b.explode_state;
      }
    }
  }
}

void draw_field( void )
{
  hline(buffer,0,BOTTOM,SCREEN_W,0x00ff00);
}


void draw_player(int x, long color)
{
  // Feet
  line(buffer,x-10,BOTTOM-1,x,BOTTOM-20,color);
  line(buffer,x,BOTTOM-20,x+10,BOTTOM-1,color);
  // Body
  line(buffer,x,BOTTOM-20,x,BOTTOM-40,color);
  // Arms
  line(buffer,x,BOTTOM-37,x-10,BOTTOM-25,color);
  line(buffer,x,BOTTOM-37,x+10,BOTTOM-25,color);
  if (!game_over)
  {
    // Head
    circle(buffer,x,BOTTOM-50,10,color);
    // Eyes
    circlefill(buffer,x-3,BOTTOM-53,2,color);
    circlefill(buffer,x+3,BOTTOM-53,2,color);
    // Mouth
    line(buffer,x-3,BOTTOM-46,x,BOTTOM-43,color);
    line(buffer,x,BOTTOM-43,x+3,BOTTOM-46,color);
  }
  else
  {
    circlefill(buffer,x,BOTTOM-50,10,color);
  }
}

void draw_timer( int seconds )
{
  textprintf_ex(buffer,font,2,2,0xffffff,-1,"Time: %d", seconds);
}
```

// Listing continued on next page...

// Listing continued from previous page

```
void draw_game_over( void )
{
  textout_centre_ex(buffer,font,"GAME OVER",SCREEN_W/2,SCREEN_H/2-
4,0xff0000,0x00000);
}

void draw_title_screen( void )
{
  const char titletxt[] = "Bomb Rain";
  BITMAP* title = create_bitmap(text_length(font,titletxt),text_height(font));
  clear(title);
  textout_ex(title,font,titletxt,0,0,0xffff00,-1);
  int width = title->w * 4;
  int height = title->h * 4;
  stretch_blit(title,buffer,0,0,title->w,title->h,(SCREEN_W-width)/2,(SCREEN_H-
height)/2,width,height);
  textout_centre_ex(buffer,font,"Created by Simon Parzer for MinorHack
2006/10/14",SCREEN_W/2,SCREEN_H/2+40,0x0000ff,-1);
  destroy_bitmap(title);
}

const int MOVEMENT = 3;

void main_loop( void )
{
  clear_music();
  add_music("c.c.c.c.g.g.g.g.e.e.e.e.c.c.c   "
            ".c.c.c.c.g.g.g.g.e.e.e.e.c       .",
            "                                  "
            "f      .c      .a      .f        .",
            "c       g        g      c         "
            "c       g       g       c        .",
            "                                  "
            "c.  c.c.c.  c.c.c.  c.c.c.   c  .");
  draw_title_screen();
  while (!keypressed()) {
    while (sC)
    {
      update_music();
      sC--;
    }
    swap_buffers();
    while (!sC)
    {
      rest(1);
    }
  }
  clear_music();
  // --------|---|---|---|---|---|---|---|---
  add_music("c C e C c C e C cCeCcCeCcCeCcCeC",
            ".                               ",
           "c      .e      .c       .e      ",
           ".                               ");

  // --------|---|---|---|---|---|---|---|---
  add_music("c  .c.c.c  .c   C  .C.C.C.  C.  ",
            "f  .f.f.f  .f   F  .F.F.F.  F.  ",
           ".                                ",
           "c.  c.  c.  c.  c.  c.  c.  c.  ");

  // --------|---|---|---|---|---|---|---|---
  add_music("c.C.d.D.e.D.d.C.c.C.d.D.e.D.d.C.",
```

// Listing continued on next page...

// Listing continued from previous page

```
            "C d D e f e D d C d D e f e D d ",
            "e   f   e   f   e   f   e   f   ",
            "c.  c.c.c.  c.  c.  c.  c.c.c.c.");

  int difficulty = 50;
  long time = 0;
  int bomb_drop = 0;
  bool running = true;
  while (running)
  {
    while (sC)
    {
      // Do logic

      if (!game_over) ++time;
      if (time%731 == 0)
      {
       --difficulty;
       if (difficulty>5) difficulty = 5;
      }

      ++bomb_drop;
      if (bomb_drop > difficulty)
      {
       bomb_drop = 0;
       add_random_bomb();
      }
      if (!game_over)
      {

       // Input
       if (key[KEY_LEFT])
       {
         player_x -= MOVEMENT;
         if (player_x < 5)
         {
           player_x += MOVEMENT;
         }
       }
       if (key[KEY_RIGHT])
       {
         player_x += MOVEMENT;
         if (player_x > SCREEN_W-5)
         {
           player_x -= MOVEMENT;
         }
       }
      }
      advance_bombs();
      if (key[KEY_ESC]) running = 0;

      // Play music
      update_music();

      // Decrease counter
      --sC;
    }
    clear(buffer);
    draw_field();
    if (!game_over)
    {
      draw_player(player_x,0x8080ff);
    }
```

// Listing continued on next page...

// Listing continued from previous page

```
    else
    {
      draw_player(player_x,0xff2020);
    }
    draw_bombs();
    draw_timer(time/FRAME_RATE);
    if (game_over) draw_game_over();
    swap_buffers();
    while (!sC)
    {
      rest(1); // Idle
    }
  }
}

int main(int argc, char** argv)
{
  srand(time(0));
  allegro_init();
  install_timer();
  install_sound(DIGI_AUTODETECT,MIDI_NONE,0);
  install_mouse();
  install_keyboard();
  set_color_depth(32);
  set_gfx_mode(GFX_AUTODETECT_WINDOWED,640,480,0,0);
  clear(screen);
  buffer=create_bitmap(SCREEN_W,SCREEN_H);
  clear(buffer);

  init_samples();
  Music.pos = 0;

  install_int(sC_int,1000/FRAME_RATE);
  main_loop();
  remove_int(sC_int);

  destroy_samples();
  destroy_bitmap(buffer);

  allegro_exit();
  return 0;
}
END_OF_MAIN();
```