NES Snake

Generated by Doxygen 1.8.12

Contents

Chapter 1

README

NES Snake This project is my first attempt to write a simple NES Snake game using Shiru´s NESLibrary, based on the CC65 project. You can find out more about Shiru´s NESLibrary here: http://shiru.untergrund.net/articles/programming_nes_games_in_c.htm Also, if you are interested in the general CC65 project, you can find it here: http://www.cc65.org/ Or just visit the project directly on GitHub: https://github.com/cc65/cc65

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 items_struct Struct Reference

This structure contains all elements required to interact with and display items.

```
#include <structures.h>
```

Data Fields

- unsigned int item_respawn_frm_rate
- unsigned char item_coordinates [ITEM_MAX_ON_SCREEN<< 1]
- unsigned int item_respawn_count [ITEM_MAX_ON_SCREEN]
- unsigned char item_collision_flags [ITEM_MAX_ON_SCREEN]

4.1.1 Detailed Description

This structure contains all elements required to interact with and display items.

Author

Sebastian Dine

4.1.2 Field Documentation

4.1.2.1 item_collision_flags

```
unsigned char item_collision_flags[ITEM_MAX_ON_SCREEN]
```

Array which indicates collisions with items (0 = no collision, 1 = collision). E.g. [0]=1 means, that the snake collided with the first element.

4.1.2.2 item_coordinates

```
unsigned char item_coordinates[ITEM_MAX_ON_SCREEN<< 1]
```

Array of item coordinates (pixel based) two elements are a coordinate set, eg. [0] is the x-coordinate of the first item and [1] its y-coordinate.

4.1.2.3 item_respawn_count

```
unsigned int item_respawn_count[ITEM_MAX_ON_SCREEN]
```

Array which counts down the frames until an item respawn. 0 indicates a non active visible item. Due to this decision, items will respawn, once their frame countdown reaches 1.

4.1.2.4 item_respawn_frm_rate

```
unsigned int item_respawn_frm_rate
```

Variable, which contains the frame rate until an items should respawn. Remember NES is 60FPS.

The documentation for this struct was generated from the following file:

- C:/Users/Administrator/Documents/GitHub/NES-Snake/src/structures.h

4.2 snake_struct Struct Reference

This structure contains all elements required to interact and display the snake.

```
#include <structures.h>
```

Data Fields

- unsigned char size_index
- unsigned char speed_counter
- unsigned char moving_direction
- unsigned char head_sprite
- unsigned char head_sprite_attribute
- unsigned char head_sprite_x
- unsigned char head_sprite_y
- unsigned char last_body_element_x
- unsigned char last_body_element_y
- unsigned char body_element_coordinates [SNAKE_MAX_SIZE<< 1]

4.2.1 Detailed Description

This structure contains all elements required to interact and display the snake.

Author

Sebastian Dine

4.2.2 Field Documentation

4.2.2.1 body_element_coordinates

```
unsigned char body_element_coordinates[SNAKE_MAX_SIZE<< 1]
```

Array of snakes body-coordinates (pixelbased), two elements are a coordinate set, eg. [0] is the x-coordinate of the first body-element and [1] its y-coordinate.

4.2.2.2 head_sprite

```
unsigned char head_sprite
```

tbd.

4.2.2.3 head_sprite_attribute

```
unsigned char head_sprite_attribute
```

Variable for holding attributes of the head sprite of the snake.

4.2.2.4 head_sprite_x

```
unsigned char head_sprite_x
```

Pixel based X-coordinate of snake's head sprite.

4.2.2.5 head_sprite_y

```
unsigned char head_sprite_y
```

Pixel based Y-coordinate of snake's head sprite.

4.2.2.6 last_body_element_x

```
unsigned char last_body_element_x
```

Pixel based X-coordinate of the last body element from last frame.

4.2.2.7 last_body_element_y

```
unsigned char last_body_element_y
```

Pixel based Y-coordinate of the last body element from last frame.

4.2.2.8 moving_direction

```
unsigned char moving_direction
```

Indicator to which direction the snake is moving. 1=up,2=down,3=left,4=right.

4.2.2.9 size_index

```
unsigned char size_index
```

Index for array 'body_element_ coordinates' which points to the space for the next body-element to add. It will be increased in +=2-steps so it always points to a free x-coordinate.

4.2.2.10 speed_counter

```
unsigned char speed_counter
```

tbd.

The documentation for this struct was generated from the following file:

- C:/Users/Administrator/Documents/GitHub/NES-Snake/src/structures.h

Chapter 5

File Documentation

5.1 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/game_over_nam.h File Reference

This header file contains the nametable (background) of the gameover screen. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Variables

- const unsigned char game_over_nam [59]

5.1.1 Detailed Description

This header file contains the nametable (background) of the gameover screen. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Author

Sebastian Dine

5.1.2 Variable Documentation

5.1.2.1 game_over_nam

```
const unsigned char game_over_nam[59]
```

Initial value:

```
={
0x01,0x00,0x01,0xe9,0x27,0x21,0x2d,0x25,0x00,0x2f,0x36,0x25,0x32,0x00,0x01,0x56,
0x33,0x23,0x2f,0x32,0x25,0x1a,0x00,0x01,0x54,0x30,0x32,0x25,0x33,0x33,0x00,0x33,
0x34,0x21,0x32,0x34,0x00,0x34,0x2f,0x00,0x23,0x2f,0x2e,0x34,0x29,0x2e,0x35,0x25,
0x00,0x01,0xfe,0x00,0x01,0xfe,0x00,0x01,0x45,0x01,0x00
}
```

5.2 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/level1_nam.h File Reference

This header file contains the nametable (background) of level map 1. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Variables

- const unsigned char level1_nam [171]

5.2.1 Detailed Description

This header file contains the nametable (background) of level map 1. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Author

Sebastian Dine

5.2.2 Variable Documentation

5.2.2.1 level1_nam

```
const unsigned char level1_nam[171]
```

Initial value:

```
={
0x01,0x00,0x01,0x20,0x33,0x23,0x2f,0x32,0x25,0x1a,0x00,0x01,0x38,0x43,0x01,0x3d,
0x44,0x44,0x43,0x43,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,
0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,
0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,
0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,
0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,
0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,
0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,
0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,
0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x2e,0x44,0x43,
0x01,0x05,0x44,0x43,0x01,0x0a,0x00,0x01,0x3f,0x01,0x00
}
```

5.3 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/level2_nam.h File Reference

This header file contains the nametable (background) of level map 2. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Variables

- const unsigned char level2_nam [264]

5.3.1 Detailed Description

This header file contains the nametable (background) of level map 2. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Author

Sebastian Dine

5.3.2 Variable Documentation

5.3.2.1 level2_nam

```
const unsigned char level2_nam[264]
```

Initial value:

```
={
0x01,0x00,0x01,0x20,0x33,0x23,0x2f,0x32,0x25,0x1a,0x00,0x01,0x38,0x43,0x01,0x3d,
0x44,0x44,0x43,0x43,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,
0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,
0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,
0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,
0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,
0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,
0x0b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,
0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,
0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,0x1b,0x43,0x01,0x03,0x00,0x01,
0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,
0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,
0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,
0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,
0x00,0x01,0x0b,0x43,0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,
0x01,0x03,0x00,0x01,0x0c,0x44,0x43,0x44,0x00,0x01,0x0b,0x43,0x01,0x2e,0x44,0x43,
0x01,0x05,0x44,0x43,0x01,0x0a,0x01,0x00
}
```

5.4 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/levels_pal.h File Reference

This header file contains the color palette for all level maps. Created with NES Screen Tool 2.04 (Option Palettes -> Put C data to clipboard.

Variables

- const unsigned char levels_pal [16]

5.4.1 Detailed Description

This header file contains the color palette for all level maps. Created with NES Screen Tool 2.04 (Option Palettes -> Put C data to clipboard.

Author

Sebastian Dine

5.4.2 Variable Documentation

5.4.2.1 levels_pal

```
const unsigned char levels_pal[16]
```

Initial value:

```
={
    0x0f,0x00,0x10,0x2a,
    0x0f,0x01,0x21,0x31,
    0x0f,0x06,0x16,0x26,
    0x0f,0x09,0x19,0x29 }
```

5.5 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/menue_pal.h File Reference

This header file contains the color palette for menus (titlescreen, gameover screen). Created with NES Screen Tool 2.04 (Option Palettes -> Put C data to clipboard.

Variables

- const unsigned char menue_pal [16]

5.5.1 Detailed Description

This header file contains the color palette for menus (titlescreen, gameover screen). Created with NES Screen Tool 2.04 (Option Palettes -> Put C data to clipboard.

Author

Sebastian Dine

5.5.2 Variable Documentation

5.5.2.1 menue_pal

```
const unsigned char menue_pal[16]
```

Initial value:

```
={
    0x0f,0x2a,0x10,0x20,
    0x0f,0x01,0x21,0x31,
    0x0f,0x06,0x16,0x26,
    0x0f,0x09,0x19,0x29 }
```

5.6 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/sprites_pal.h File Reference

This header file contains the color palette for sprites.

Variables

- const unsigned char sprites_pal [16]

5.6.1 Detailed Description

This header file contains the color palette for sprites.

Author

Sebastian Dine

5.6.2 Variable Documentation

5.6.2.1 sprites_pal

```
const unsigned char sprites_pal[16]
```

Initial value:

```
={
        0x0f,0x17,0x27,0x37,
        0x0f,0x11,0x21,0x31,
        0x0f,0x15,0x25,0x35,
        0x0f,0x19,0x29,0x2a }
```

5.7 C:/Users/Administrator/Documents/GitHub/NES-Snake/gfx/titlescreen_nam.h File Reference

This header file contains the nametable (background) of the titlescreen. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Variables

- const unsigned char titlescreen_nam [253]

5.7.1 Detailed Description

This header file contains the nametable (background) of the titlescreen. Created with NES Screen Tool 2.04 (Option Nametable -> Save nametable and attributes -> RLE packed as C header (.h).

Author

Sebastian Dine

5.7.2 Variable Documentation

5.7.2.1 titlescreen_nam

```
const unsigned char titlescreen_nam[253]
```

Initial value:

```
={
0x01,0x43,0x01,0x3f,0x44,0x44,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,
0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x06,0x50,0x51,0x52,0x53,0x54,
0x55,0x50,0x51,0x56,0x57,0x58,0x59,0x52,0x53,0x00,0x01,0x06,0x44,0x01,0x03,0x00,
0x01,0x06,0x60,0x61,0x62,0x63,0x64,0x65,0x60,0x61,0x66,0x67,0x68,0x69,0x62,0x63,
0x00,0x01,0x06,0x44,0x01,0x03,0x00,0x01,0x06,0x70,0x71,0x72,0x73,0x74,0x75,0x70,
0x71,0x76,0x77,0x78,0x79,0x72,0x73,0x00,0x01,0x06,0x44,0x01,0x03,0x00,0x01,0x1b,
0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,
0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x07,0x30,0x32,0x25,0x33,0x33,0x00,0x33,0x34,
0x21,0x32,0x34,0x00,0x01,0x08,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,
0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,
0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,
0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,
0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,
0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x00,0x01,0x1b,0x44,0x01,0x03,0x33,0x25,0x22,
0x21,0x33,0x34,0x29,0x21,0x2e,0x00,0x24,0x29,0x2e,0x25,0x0c,0x12,0x10,0x11,0x16,
0x00,0x01,0x08,0x44,0x44,0x43,0x01,0x3f,0x00,0x01,0x3f,0x01,0x00
}
```

5.8 C:/Users/Administrator/Documents/GitHub/NES-Snake/NESLibrary/neslib.h File Reference

Macros

- #define PAD_A 0x01
- #define PAD_B 0x02
- #define PAD_SELECT 0x04
- #define PAD_START 0x08
- #define PAD_UP 0x10
- #define PAD_DOWN 0x20
- #define PAD_LEFT 0x40
- #define PAD_RIGHT 0x80
- #define OAM_FLIP_V 0x80
- #define OAM_FLIP_H 0x40
- #define OAM_BEHIND 0x20
- #define MAX(x1, x2) ((x1)<(x2)?(x2):(x1))
- #define MIN(x1, x2) ((x1)<(x2)?(x1):(x2))
- #define MASK_SPR 0x10
- #define MASK_BG 0x08
- #define MASK_EDGE_SPR 0x04
- #define MASK_EDGE_BG 0x02
- #define NAMETABLE_A 0x2000
- #define NAMETABLE_B 0x2400
- #define NAMETABLE_C 0x2800
- #define NAMETABLE_D 0x2c00
- #define NULL 0
- #define TRUE 1
- #define FALSE 0
- #define NT_UPD_HORZ 0x40
- #define NT_UPD_VERT 0x80
- #define NT_UPD_EOF 0xff
- #define NTADR_A(x, y) (NAMETABLE_A|(((y)<<5)|(x)))
- #define NTADR_B(x, y) (NAMETABLE_B|(((y)<<5)|(x)))
- #define NTADR_C(x, y) (NAMETABLE_C|(((y)<<5)|(x)))
- #define NTADR_D(x, y) (NAMETABLE_D|(((y)<<5)|(x)))
- #define MSB(x) (((x)>>8))

Functions

- void __fastcall__ pal_all (const char ∗data)
- void __fastcall__ pal_bg (const char ∗data)
- void __fastcall__ pal_spr (const char ∗data)
- void __fastcall__ pal_col (unsigned char index, unsigned char color)
- void __fastcall__ pal_clear (void)
- void __fastcall__ pal_bright (unsigned char bright)
- void __fastcall__ pal_spr_bright (unsigned char bright)
- void __fastcall__ pal_bg_bright (unsigned char bright)
- void __fastcall__ ppu_wait_nmi (void)
- void __fastcall__ ppu_wait_frame (void)
- void __fastcall__ ppu_off (void)
- void __fastcall__ ppu_on_all (void)
- void __fastcall__ ppu_on_bg (void)
- void __fastcall__ ppu_on_spr (void)
- void __fastcall__ ppu_mask (unsigned char mask)
- unsigned char __fastcall__ ppu_system (void)
- void __fastcall__ oam_clear (void)
- void __fastcall__ oam_size (unsigned char size)
- unsigned char __fastcall__ oam_spr (unsigned char x, unsigned char y, unsigned char chrnum, unsigned char attr, unsigned char sprid)
- unsigned char __fastcall__ oam_meta_spr (unsigned char x, unsigned char y, unsigned char sprid, const unsigned char ∗data)
- void __fastcall__ oam_hide_rest (unsigned char sprid)
- void __fastcall__ music_play (unsigned char song)
- void __fastcall__ music_stop (void)
- void __fastcall__ music_pause (unsigned char pause)
- void __fastcall__ sfx_play (unsigned char sound, unsigned char channel)
- void __fastcall__ sample_play (unsigned char sample)
- unsigned char __fastcall__ pad_poll (unsigned char pad)
- unsigned char __fastcall__ pad_trigger (unsigned char pad)
- unsigned char __fastcall__ pad_state (unsigned char pad)
- void __fastcall__ scroll (unsigned int x, unsigned int y)
- void __fastcall__ split (unsigned int x, unsigned int y)
- void __fastcall__ bank_spr (unsigned char n)
- void __fastcall__ bank_bg (unsigned char n)
- unsigned char __fastcall__ rand8 (void)
- unsigned int __fastcall__ rand16 (void)
- void __fastcall__ set_rand (unsigned int seed)
- void __fastcall__ set_vram_update (unsigned char ∗buf)
- void __fastcall__ flush_vram_update (unsigned char ∗buf)
- void __fastcall__ vram_adr (unsigned int adr)
- void __fastcall__ vram_put (unsigned char n)
- void __fastcall__ vram_fill (unsigned char n, unsigned int len)
- void __fastcall__ vram_inc (unsigned char n)
- void __fastcall__ vram_read (unsigned char ∗dst, unsigned int size)
- void __fastcall__ vram_write (unsigned char ∗src, unsigned int size)
- void __fastcall__ vram_unrle (const unsigned char ∗data)
- void __fastcall__ memcpy (void ∗dst, void ∗src, unsigned int len)
- void __fastcall__ memfill (void ∗dst, unsigned char value, unsigned int len)
- void __fastcall__ delay (unsigned char frames)

5.8.1 Macro Definition Documentation

5.8.1.1 FALSE

```
#define FALSE 0
```

5.8.1.2 MASK_BG

```
#define MASK_BG 0x08
```

5.8.1.3 MASK_EDGE_BG

```
#define MASK_EDGE_BG 0x02
```

5.8.1.4 MASK_EDGE_SPR

```
#define MASK_EDGE_SPR 0x04
```

5.8.1.5 MASK_SPR

```
#define MASK_SPR 0x10
```

5.8.1.6 MAX

```
#define MAX(
              x1,
              x2 ) ((x1)<(x2)?(x2):(x1))
```

5.8.1.7 MIN

```
#define MIN(
              x1,
              x2 ) ((x1)<(x2)?(x1):(x2))
```

5.8.1.8 MSB

```
#define MSB(
              x ) (((x)>>8))
```

5.8.1.9 NAMETABLE_A

```
#define NAMETABLE_A 0x2000
```

5.8.1.10 NAMETABLE_B

```
#define NAMETABLE_B 0x2400
```

5.8.1.11 NAMETABLE_C

```
#define NAMETABLE_C 0x2800
```

5.8.1.12 NAMETABLE_D

```
#define NAMETABLE_D 0x2c00
```

5.8.1.13 NT_UPD_EOF

```
#define NT_UPD_EOF 0xff
```

5.8.1.14 NT_UPD_HORZ

```
#define NT_UPD_HORZ 0x40
```

5.8.1.15 NT_UPD_VERT

```
#define NT_UPD_VERT 0x80
```

5.8.1.16 NTADR_A

```
#define NTADR_A(
              x,
              y ) (NAMETABLE_A|(((y)<<5)|(x)))
```

5.8.1.17 NTADR_B

```
#define NTADR_B(
              x,
              y ) (NAMETABLE_B|(((y)<<5)|(x)))
```

5.8.1.18 NTADR_C

```
#define NTADR_C(
              x,
              y ) (NAMETABLE_C|(((y)<<5)|(x)))
```

5.8.1.19 NTADR_D

```
#define NTADR_D(
              x,
              y ) (NAMETABLE_D|(((y)<<5)|(x)))
```

5.8.1.20 NULL

```
#define NULL 0
```

5.8.1.21 OAM_BEHIND

```
#define OAM_BEHIND 0x20
```

5.8.1.22 OAM_FLIP_H

```
#define OAM_FLIP_H 0x40
```

5.8.1.23 OAM_FLIP_V

```
#define OAM_FLIP_V 0x80
```

5.8.1.24 PAD_A

```
#define PAD_A 0x01
```

5.8.1.25 PAD_B

```
#define PAD_B 0x02
```

5.8.1.26 PAD_DOWN

```
#define PAD_DOWN 0x20
```

5.8.1.27 PAD_LEFT

```
#define PAD_LEFT 0x40
```

5.8.1.28 PAD_RIGHT

```
#define PAD_RIGHT 0x80
```

5.8.1.29 PAD_SELECT

```
#define PAD_SELECT 0x04
```

5.8.1.30 PAD_START

```
#define PAD_START 0x08
```

5.8.1.31 PAD_UP

```
#define PAD_UP 0x10
```

5.8.1.32 TRUE

```
#define TRUE 1
```

5.8.2 Function Documentation

5.8.2.1 bank_bg()

```
void __fastcall__ bank_bg (
             unsigned char n )
```

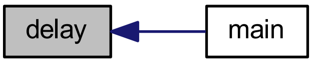
5.8.2.2 bank_spr()

```
void __fastcall__ bank_spr (
             unsigned char n )
```

5.8.2.3 delay()

```
void __fastcall__ delay (
             unsigned char frames )
```

Here is the caller graph for this function:



5.8.2.4 flush_vram_update()

```
void __fastcall__ flush_vram_update (
            unsigned char * buf )
```

5.8.2.5 memcpy()

```
void __fastcall__ memcpy (
            void * dst,
            void * src,
            unsigned int len )
```

5.8.2.6 memfill()

```
void __fastcall__ memfill (
            void * dst,
            unsigned char value,
            unsigned int len )
```

5.8.2.7 music_pause()

```
void __fastcall__ music_pause (
            unsigned char pause )
```

5.8.2.8 music_play()

```
void __fastcall__ music_play (
            unsigned char song )
```

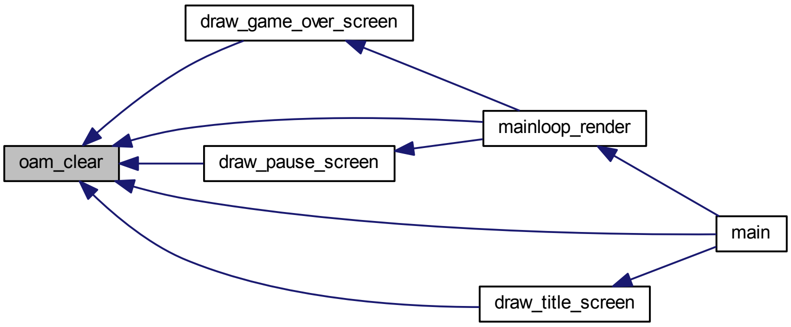
5.8.2.9 music_stop()

```
void __fastcall__ music_stop (
            void )
```

5.8.2.10 oam_clear()

```
void __fastcall__ oam_clear (
             void  )
```

Here is the caller graph for this function:



5.8.2.11 oam_hide_rest()

```
void __fastcall__ oam_hide_rest (
             unsigned char sprid )
```

5.8.2.12 oam_meta_spr()

```
unsigned char __fastcall__ oam_meta_spr (
             unsigned char x,
             unsigned char y,
             unsigned char sprid,
             const unsigned char * data )
```

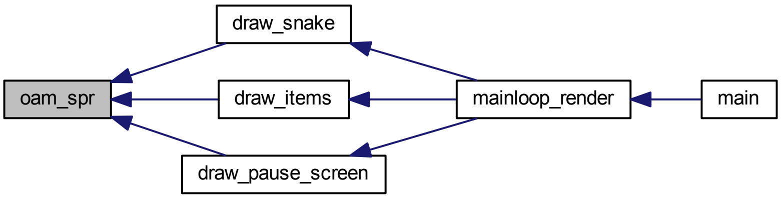
5.8.2.13 oam_size()

```
void __fastcall__ oam_size (
             unsigned char size )
```

5.8.2.14 oam_spr()

```
unsigned char __fastcall__ oam_spr (
            unsigned char x,
            unsigned char y,
            unsigned char chrnum,
            unsigned char attr,
            unsigned char sprid )
```

Here is the caller graph for this function:



5.8.2.15 pad_poll()

```
unsigned char __fastcall__ pad_poll (
            unsigned char pad )
```

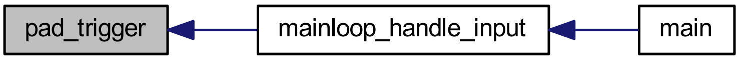
5.8.2.16 pad_state()

```
unsigned char __fastcall__ pad_state (
            unsigned char pad )
```

5.8.2.17 pad_trigger()

```
unsigned char __fastcall__ pad_trigger (
            unsigned char pad )
```

Here is the caller graph for this function:



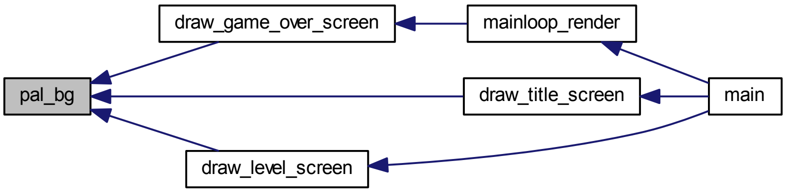
5.8.2.18 pal_all()

```
void __fastcall__ pal_all (
            const char * data )
```

5.8.2.19 pal_bg()

```
void __fastcall__ pal_bg (
            const char * data )
```

Here is the caller graph for this function:



5.8.2.20 pal_bg_bright()

```
void __fastcall__ pal_bg_bright (
            unsigned char bright )
```

5.8.2.21 pal_bright()

```
void __fastcall__ pal_bright (
            unsigned char bright )
```

5.8.2.22 pal_clear()

```
void __fastcall__ pal_clear (
            void  )
```

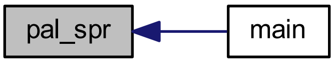
5.8.2.23 pal_col()

```
void __fastcall__ pal_col (
            unsigned char index,
            unsigned char color )
```

5.8.2.24 pal_spr()

```
void __fastcall__ pal_spr (
            const char * data )
```

Here is the caller graph for this function:



5.8.2.25 pal_spr_bright()

```
void __fastcall__ pal_spr_bright (
            unsigned char bright )
```

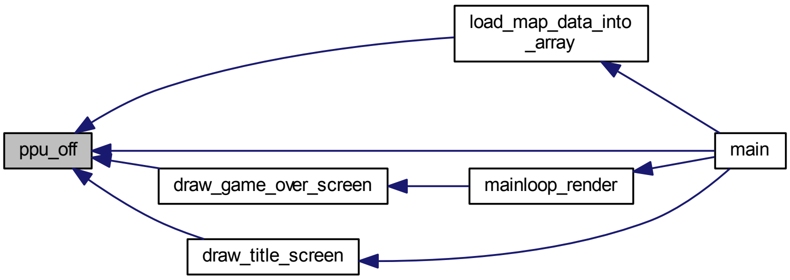
5.8.2.26 ppu_mask()

```
void __fastcall__ ppu_mask (
            unsigned char mask )
```

5.8.2.27 ppu_off()

```
void __fastcall__ ppu_off (
            void  )
```

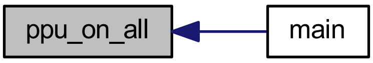
Here is the caller graph for this function:



5.8.2.28 ppu_on_all()

```
void __fastcall__ ppu_on_all (
             void )
```

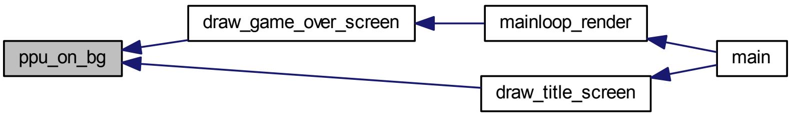
Here is the caller graph for this function:



5.8.2.29 ppu_on_bg()

```
void __fastcall__ ppu_on_bg (
             void )
```

Here is the caller graph for this function:



5.8.2.30 ppu_on_spr()

```
void __fastcall__ ppu_on_spr (
             void )
```

5.8.2.31 ppu_system()

```
unsigned char __fastcall__ ppu_system (
             void )
```

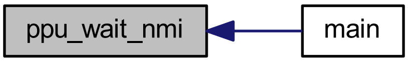
5.8.2.32 ppu_wait_frame()

```
void __fastcall__ ppu_wait_frame (
             void )
```

5.8.2.33 ppu_wait_nmi()

```
void __fastcall__ ppu_wait_nmi (
             void  )
```

Here is the caller graph for this function:



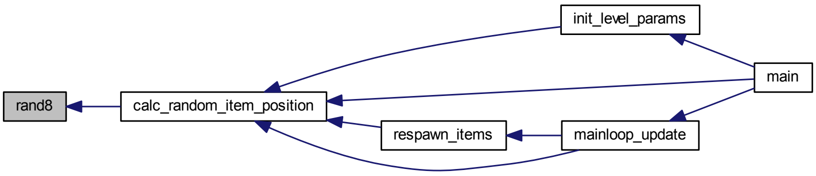
5.8.2.34 rand16()

```
unsigned int __fastcall__ rand16 (
             void  )
```

5.8.2.35 rand8()

```
unsigned char __fastcall__ rand8 (
             void  )
```

Here is the caller graph for this function:



5.8.2.36 sample_play()

```
void __fastcall__ sample_play (
             unsigned char sample )
```

5.8.2.37 scroll()

```
void __fastcall__ scroll (
             unsigned int x,
             unsigned int y )
```

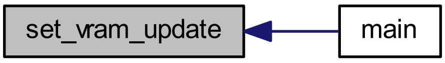
5.8.2.38 set_rand()

```
void __fastcall__ set_rand (
            unsigned int seed )
```

5.8.2.39 set_vram_update()

```
void __fastcall__ set_vram_update (
            unsigned char * buf )
```

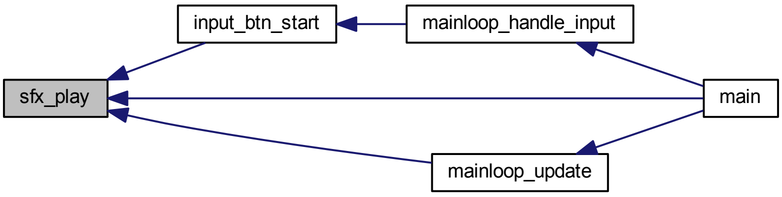
Here is the caller graph for this function:



5.8.2.40 sfx_play()

```
void __fastcall__ sfx_play (
            unsigned char sound,
            unsigned char channel )
```

Here is the caller graph for this function:



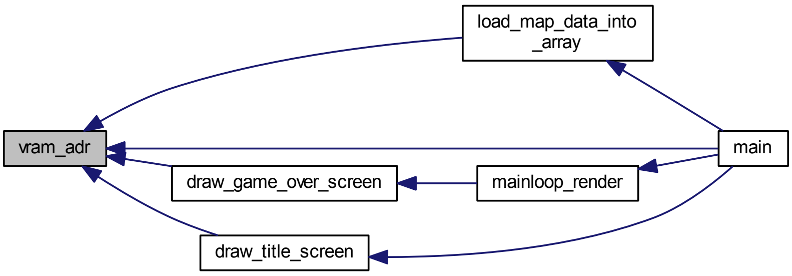
5.8.2.41 split()

```
void __fastcall__ split (
            unsigned int x,
            unsigned int y )
```

5.8.2.42 vram_adr()

```
void __fastcall__ vram_adr (
            unsigned int adr )
```

Here is the caller graph for this function:



5.8.2.43 vram_fill()

```
void __fastcall__ vram_fill (
            unsigned char n,
            unsigned int len )
```

5.8.2.44 vram_inc()

```
void __fastcall__ vram_inc (
            unsigned char n )
```

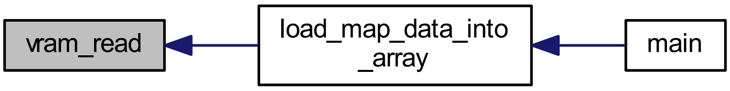
5.8.2.45 vram_put()

```
void __fastcall__ vram_put (
            unsigned char n )
```

5.8.2.46 vram_read()

```
void __fastcall__ vram_read (
            unsigned char * dst,
            unsigned int size )
```

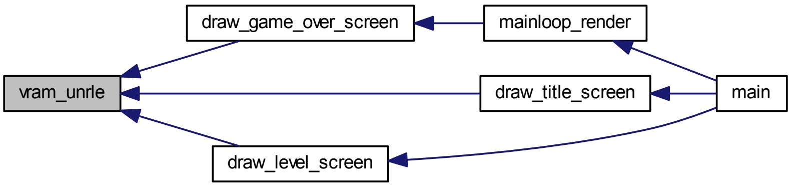
Here is the caller graph for this function:



5.8.2.47 vram_unrle()

```
void __fastcall__ vram_unrle (
            const unsigned char * data )
```

Here is the caller graph for this function:



5.8.2.48 vram_write()

```
void __fastcall__ vram_write (
            unsigned char * src,
            unsigned int size )
```

5.9 C:/Users/Administrator/Documents/GitHub/NES-Snake/README.md File Reference

5.10 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/globals.h File Reference

This header file defines all global variables of the game.

Variables

- static struct snake_struct snake
- static struct items_struct items

Global variables, which are used for several coordination calculations,

e.g. calculate pixel based coordinates (of body elements) to tile based coordinates.

- static unsigned char coord_x
- static unsigned char coord_y

Global variables, used to modify the background ingame

- static unsigned char update_list [5 ∗3+1]
- static unsigned char ∗ ul

Global variables, used for rendering sprites ingame

- static unsigned char sprite_offset

Global variables, used for universal purpose e.g loops

- static unsigned char i
- static unsigned char j
- static unsigned int k
- static unsigned int l

Global variables, used for game-states, menues, input

- static unsigned char current_level
- static unsigned char max_score
- static unsigned char pause
- static unsigned char gameover
- static unsigned char input
- static unsigned char pause_loop
- static unsigned char gameover_loop
- static unsigned char titlescreen
- static unsigned char restart

Global variables, used to interact with the level map

- static unsigned char map [MAP_WIDTH ∗MAP_HEIGHT]
- static unsigned char nameRow [MAP_WIDTH]
- static unsigned int nametable_fetch

List of the levels, include pointer to the packed nametable of the levels, menues, and pointer to the associated palette.

- const unsigned char ∗const levelList [LEVELS_ALL+2+2]

5.10.1 Detailed Description

This header file defines all global variables of the game.

Author

Sebastian Dine

5.10.2 Variable Documentation

5.10.2.1 coord_x

```
unsigned char coord_x  [static]
```

5.10.2.2 coord_y

```
unsigned char coord_y  [static]
```

5.10.2.3 current_level

```
unsigned char current_level  [static]
```

Global variable, indicating the current level.

5.10.2.4 gameover

```
unsigned char gameover  [static]
```

Global variable, indicating the game over mode (1= game over 0= no game over).

5.10.2.5 gameover_loop

```
unsigned char gameover_loop  [static]
```

identifier to check, if first gameover loop is passed (1= true, 0= false).

5.10.2.6 i

```
unsigned char i  [static]
```

5.10.2.7 input

```
unsigned char input  [static]
```

Global variable, holding the controller input of the current frame

5.10.2.8 items

```
struct items_struct items  [static]
```

Global variable, containing all elements used to interact with and display items

5.10.2.9 j

```
unsigned char j  [static]
```

5.10.2.10 k

```
unsigned int k  [static]
```

5.10.2.11 l

```
unsigned int l  [static]
```

5.10.2.12 levelList

```
const unsigned char* const levelList[LEVELS_ALL+2+2]
```

Initial value:

```
={
    level1_nam, level2_nam,
    game_over_nam, titlescreen_nam,
    levels_pal, menue_pal
}
```

5.10.2.13 map

```
unsigned char map[MAP_WIDTH *MAP_HEIGHT]  [static]
```

Array of the complete game map (tile-based).

5.10.2.14 max_score

```
unsigned char max_score  [static]
```

Global variable, indicating the maximum score of the current level.

5.10.2.15 nameRow

```
unsigned char nameRow[MAP_WIDTH]  [static]
```

Array for fetching nametable into array 'map', row by row.

5.10.2.16 nametable_fetch

```
unsigned int nametable_fetch  [static]
```

Variable for fetching through nametable.

5.10.2.17 pause

```
unsigned char pause  [static]
```

Global variable, indicating the pause mode (1= pause, 0= no pause).

5.10.2.18 pause_loop

```
unsigned char pause_loop  [static]
```

Identifier to check, if first pause-loop is passed (1= true, 0= false).

5.10.2.19 restart

```
unsigned char restart  [static]
```

Global variable, for handling the restart input

5.10.2.20 snake

```
struct snake_struct snake  [static]
```

Global variable, containing all elements used to interact and display the snake

5.10.2.21 sprite_offset

```
unsigned char sprite_offset  [static]
```

5.10.2.22 titlescreen

```
unsigned char titlescreen  [static]
```

Global variable, indicating the titlescreen mode (1=titlescreen 0= no titlescreen).

5.10.2.23 ul

```
unsigned char* ul  [static]
```

Pointer to array 'update_list' to enable better handling of the list

5.10.2.24 update_list

```
unsigned char update_list[5 *3+1]  [static]
```

Array of bg-elements which will be used to update VRAM once per frame. Every 3 entries are describing one bg-element.

- the first 3 elements (9 array-elements) are assigned to the game score
- the 4. and 5. element are assigned to the first and last body element of the snake
- the last array-element needs to be the VRAM end-of-file-indicator NT_UPD_EOF.

Only two body elements need to be updated once per frame:

- The new first body element needs to be drawn
- The old last body element need to be disabled

5.11 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/init.c File Reference

This file contains functions for initializing game elements.

Functions

- void calc_random_item_position (void)
- void load_map_data_into_array (void)
- void init_items (void)
- void init_level_params (void)

5.11.1 Detailed Description

This file contains functions for initializing game elements.

Author

Sebastian Dine

5.11.2 Function Documentation

5.11.2.1 calc_random_item_position()

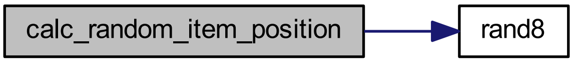
```
void calc_random_item_position (
            void  )
```

This function calculates the coordinates of an grow-item. It stores the calculated coordinates into global fields 'coord_x' and 'coord_y'. In terms of the game structure, this function should be placed in file 'update.c'. But since I would like to spawn the initial items randomly as well, this function needs to placed in file 'init.c'.

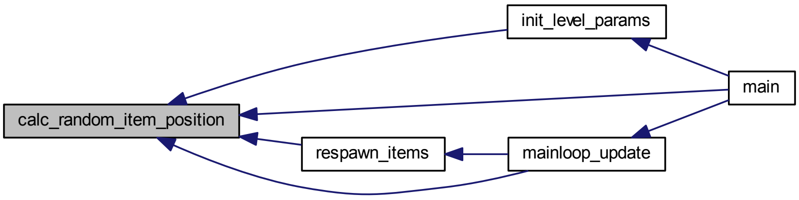
Author

Sebastian Dine

Here is the call graph for this function:



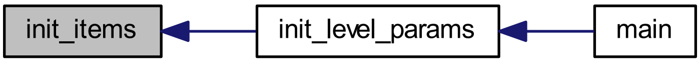
Here is the caller graph for this function:



5.11.2.2 init_items()

```
void init_items (
             void  )
```

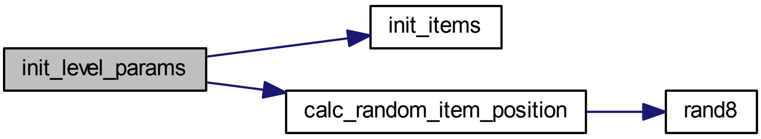
Here is the caller graph for this function:



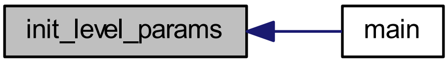
5.11.2.3 init_level_params()

```
void init_level_params (
             void  )
```

This function initializes game elements, which differ between levels. (e.g. score to reach for next level or start position of the snake) Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.4 load_map_data_into_array()

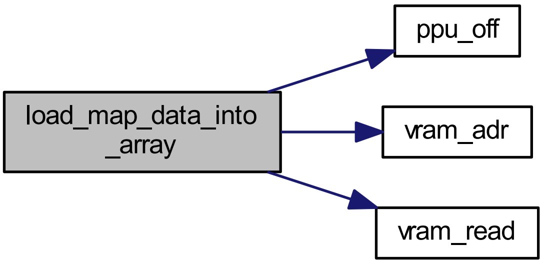
```
void load_map_data_into_array (
            void  )
```

This function reads the namespace into global array 'map', which is used for further calculations, e.g. collision detection.

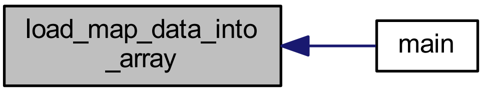
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.12 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/input.c File Reference

This file contains functions for input handling from a controller.

Functions

- void input_btn_start (void)
- void mainloop_handle_input (void)

5.12.1 Detailed Description

This file contains functions for input handling from a controller.

Author

Sebastian Dine

5.12.2 Function Documentation

5.12.2.1 input_btn_start()

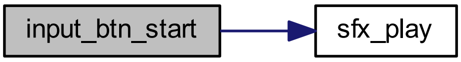
```
void input_btn_start (
            void )
```

This function contains the logic for the START button according to different scenarios e.g. title screen, ingame, gameover.

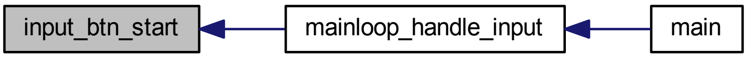
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.2 mainloop_handle_input()

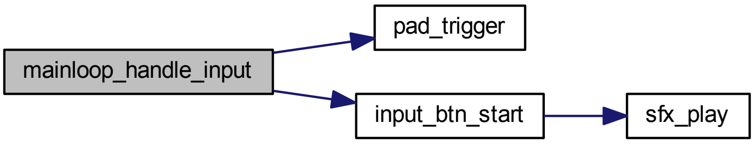
```
void mainloop_handle_input (
             void  )
```

This function provides the main input handling functionalities for an controller on port 1. It contains logic for input of the following buttons: UP, DOWN, LEFT, RIGHT, START.

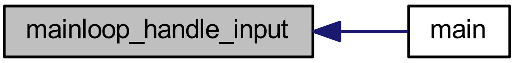
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.13 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/macros.h File Reference

This header file defines object-like macros (constants) and function-like macros for more efficient calculations.

Macros

- #define LEVELS_ALL 5
- #define SNAKE_MAX_SIZE 100
- #define ITEM_MAX_ON_SCREEN 4
- #define LVL1_START_X 120
- #define LVL1_START_Y 120
- #define LVL1_MAX_SCORE 4
- #define LVL2_START_X 56

- #define LVL2_START_Y 120
- #define LVL2_MAX_SCORE 8
- #define NAMETABLE1_START 0x2000

Tile-based width and height of the level map

- #define MAP_WIDTH 32
- #define MAP_HEIGHT 30

Direction constants

- #define DIR_UP 1
- #define DIR_DOWN 2
- #define DIR_LEFT 3
- #define DIR_RIGHT 4

Tile constants

- #define WALL_TILE_1 0x43
- #define WALL_TILE_2 0x44
- #define SNAKE_HEAD_TILE_VERT 0x41
- #define SNAKE_HEAD_TILE_HORZ 0x42
- #define SNAKE_BODY_TILE 0x40
- #define EMPTY_TILE 0x00
- #define SPIDER_TILE 0x45
- #define DIGIT_O_TILE 0x10

Sound effect constants.

Each constants represents the number of an sound effect within sfx/snake_sfx.s (connect between game and .s file is created in file NESLibrary/crt0.s).

- #define SFX_ABSORB_ITEM 0
- #define SFX_CRASH 1
- #define SFX_PAUSE 2
- #define SFX_NEXT_LEVEL 3

Macros for more efficent caluclations

- #define MAPARRAY_ADR(x, y) ((y<<2)|(x>>3))

5.13.1 Detailed Description

This header file defines object-like macros (constants) and function-like macros for more efficient calculations.

Author

Sebastian Dine

5.13.2 Macro Definition Documentation

5.13.2.1 DIGIT_O_TILE

```
#define DIGIT_O_TILE 0x10
```

Tile of digit 0 (zero)

5.13.2.2 DIR_DOWN

```
#define DIR_DOWN 2
```

5.13.2.3 DIR_LEFT

```
#define DIR_LEFT 3
```

5.13.2.4 DIR_RIGHT

```
#define DIR_RIGHT 4
```

5.13.2.5 DIR_UP

```
#define DIR_UP 1
```

5.13.2.6 EMPTY_TILE

```
#define EMPTY_TILE 0x00
```

Tile of empty space

5.13.2.7 ITEM_MAX_ON_SCREEN

```
#define ITEM_MAX_ON_SCREEN 4
```

Maximum of items, that can be on the screen on the same time.

5.13.2.8 LEVELS_ALL

```
#define LEVELS_ALL 5
```

Total number of level maps (ingame background nametables)

5.13.2.9 LVL1_MAX_SCORE

```
#define LVL1_MAX_SCORE 4
```

5.13.2.10 LVL1_START_X

```
#define LVL1_START_X 120
```

5.13.2.11 LVL1_START_Y

```
#define LVL1_START_Y 120
```

5.13.2.12 LVL2_MAX_SCORE

```
#define LVL2_MAX_SCORE 8
```

5.13.2.13 LVL2_START_X

```
#define LVL2_START_X 56
```

5.13.2.14 LVL2_START_Y

```
#define LVL2_START_Y 120
```

5.13.2.15 MAP_HEIGHT

```
#define MAP_HEIGHT 30
```

5.13.2.16 MAP_WIDTH

```
#define MAP_WIDTH 32
```

5.13.2.17 MAPARRAY_ADR

```
#define MAPARRAY_ADR(
              x,
              y ) ((y<<2)|(x>>3))
```

Macro for calculating in which tile of the 32∗30 tiles the given position is placed. Optimized with bitshifting, arithmetic pendant is (((y/8)∗32)+(x/8)). x and y are assumed to be Sprite-coordinates (not Tile-coordinates).

5.13.2.18 NAMETABLE1_START

```
#define NAMETABLE1_START 0x2000
```

Start address in VRAM for first nametable

5.13.2.19 SFX_ABSORB_ITEM

```
#define SFX_ABSORB_ITEM 0
```

5.13.2.20 SFX_CRASH

```
#define SFX_CRASH 1
```

5.13.2.21 SFX_NEXT_LEVEL

```
#define SFX_NEXT_LEVEL 3
```

5.13.2.22 SFX_PAUSE

```
#define SFX_PAUSE 2
```

5.13.2.23 SNAKE_BODY_TILE

```
#define SNAKE_BODY_TILE 0x40
```

Tile of snake body element

5.13.2.24 SNAKE_HEAD_TILE_HORZ

```
#define SNAKE_HEAD_TILE_HORZ 0x42
```

Tile of horizontal snake head element

5.13.2.25 SNAKE_HEAD_TILE_VERT

```
#define SNAKE_HEAD_TILE_VERT 0x41
```

Tile of vertical snake head element

5.13.2.26 SNAKE_MAX_SIZE

```
#define SNAKE_MAX_SIZE 100
```

Maximum of body elements, the snake can get.

5.13.2.27 SPIDER_TILE

```
#define SPIDER_TILE 0x45
```

Tile of spider item

5.13.2.28 WALL_TILE_1

```
#define WALL_TILE_1 0x43
```

Tile of horiontal wall element

5.13.2.29 WALL_TILE_2

```
#define WALL_TILE_2 0x44
```

Tile of vertical wall element

5.14 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/render.c File Reference

This file contains all functionality to draw onto the screen, eighter as sprites or as background tiles.

Functions

- void draw_snake (void)
- void draw_items (void)
- void draw_score (void)
- void init_updateList (void)
- void center_score_when_gameover (void)
- void draw_game_over_screen (void)
- void draw_title_screen (void)
- void draw_pause_screen (void)
- void draw_level_screen (void)
- void mainloop_render (void)

5.14.1 Detailed Description

This file contains all functionality to draw onto the screen, eighter as sprites or as background tiles.

Author

Sebastian Dine

5.14.2 Function Documentation

5.14.2.1 center_score_when_gameover()

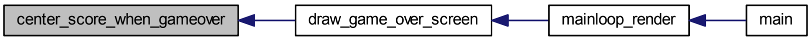
```
void center_score_when_gameover (
            void  )
```

This function moves the rendering of the score from the upper left corner to the center of the screen.

Author

Sebastian Dine

Here is the caller graph for this function:



5.14.2.2 draw_game_over_screen()

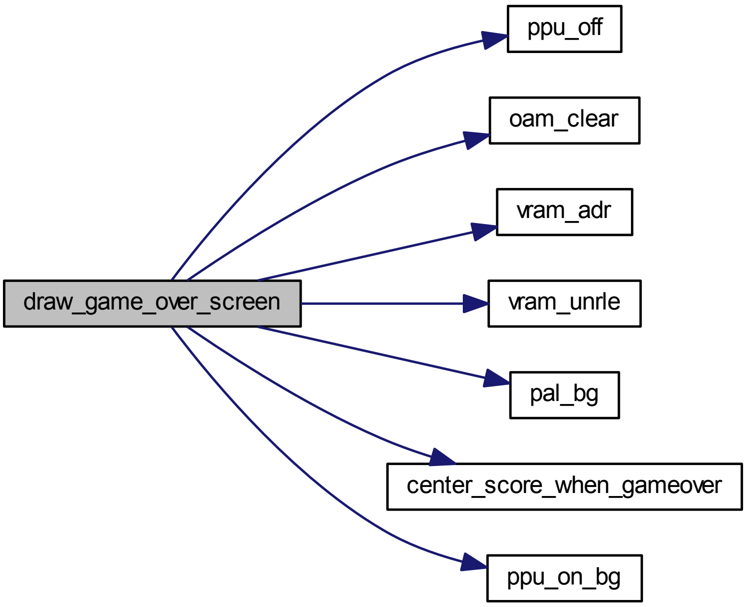
```
void draw_game_over_screen (
             void  )
```

This function draws the gameover screen.

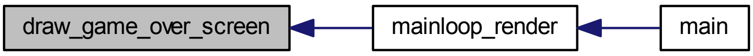
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.3 draw_items()

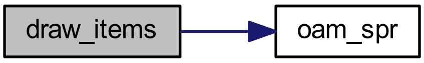
```
void draw_items (
             void  )
```

This function draws all item elements as sprites to the screen.

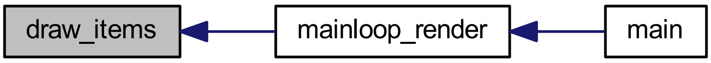
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.4 draw_level_screen()

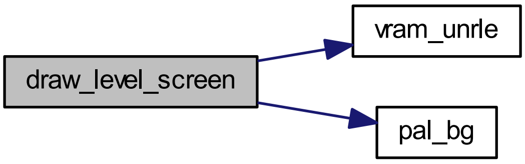
```
void draw_level_screen (
             void  )
```

This function draws the background of the current level to the screen.

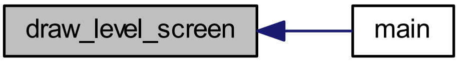
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.5 draw_pause_screen()

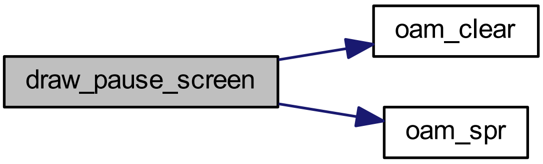
```
void draw_pause_screen (
             void  )
```

This function draws the letters PAUSE as sprites to the center of the screen, if the game is paused.

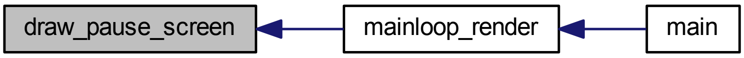
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.6 draw_score()

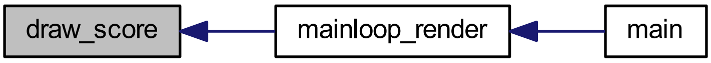
```
void draw_score (
            void  )
```

This function draws the current score as background tiles to the screen.

Author

Sebastian Dine

Here is the caller graph for this function:



5.14.2.7 draw_snake()

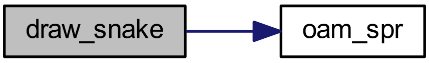
```
void draw_snake (
            void  )
```

This function draws the whole snake. The head will be drawn as a sprite, the body elements as background tiles.

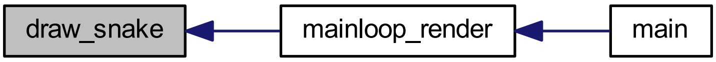
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.8 draw_title_screen()

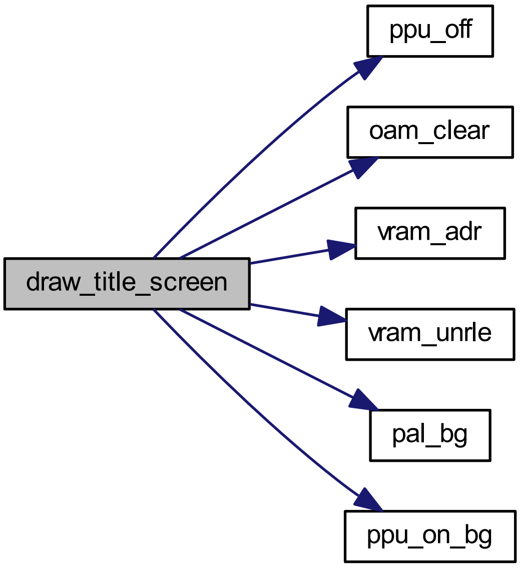
```
void draw_title_screen (
             void  )
```

This function draws the title screen.

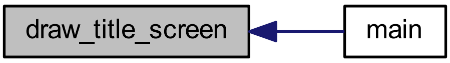
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.9 init_updateList()

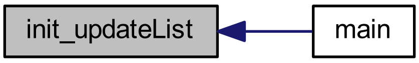
```
void init_updateList (
             void  )
```

This function initializes the (background tile) update-list with score-elements (zero-digits) and the EOF-indicator.

Author

Sebastian Dine

Here is the caller graph for this function:



5.14.2.10 mainloop_render()

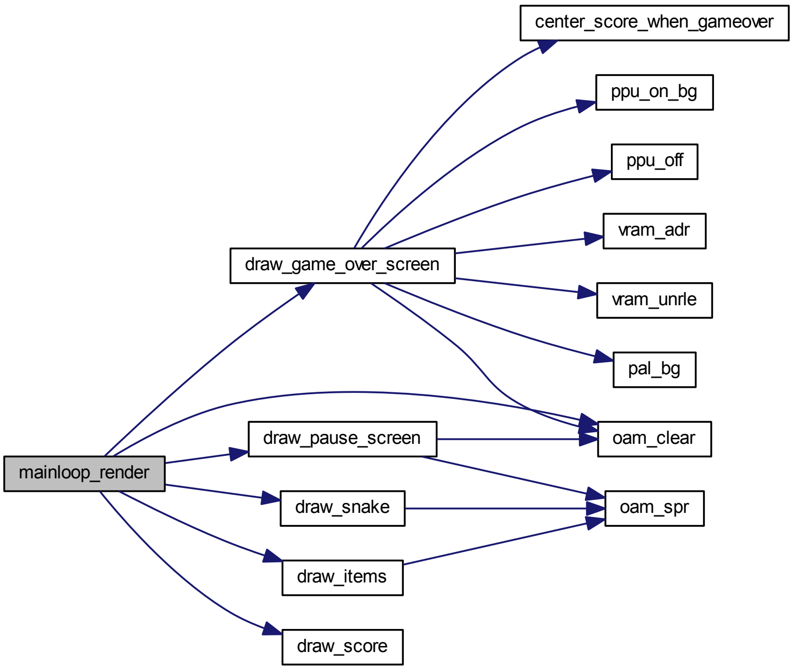
```
void mainloop_render (
            void  )
```

This function provides the coordination of all render routines according to the current status of the game, once per frame.

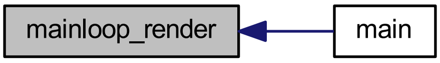
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.15 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/snake.c File Reference

Maingame file, containing the main game loop.

```
#include "level1_nam.h"
#include "level2_nam.h"
#include "game_over_nam.h"
#include "titlescreen_nam.h"
#include "levels_pal.h"
#include "sprites_pal.h"
#include "menue_pal.h"
#include "neslib.h"
#include "macros.h"
#include "structures.h"
#include "globals.h"
#include "init.c"
#include "input.c"
#include "update.c"
#include "render.c"
```

Functions

- void main (void)

  Main game loop.

5.15.1 Detailed Description

Maingame file, containing the main game loop.

Author

Sebastian Dine.

5.15.2 Function Documentation

5.15.2.1 main()

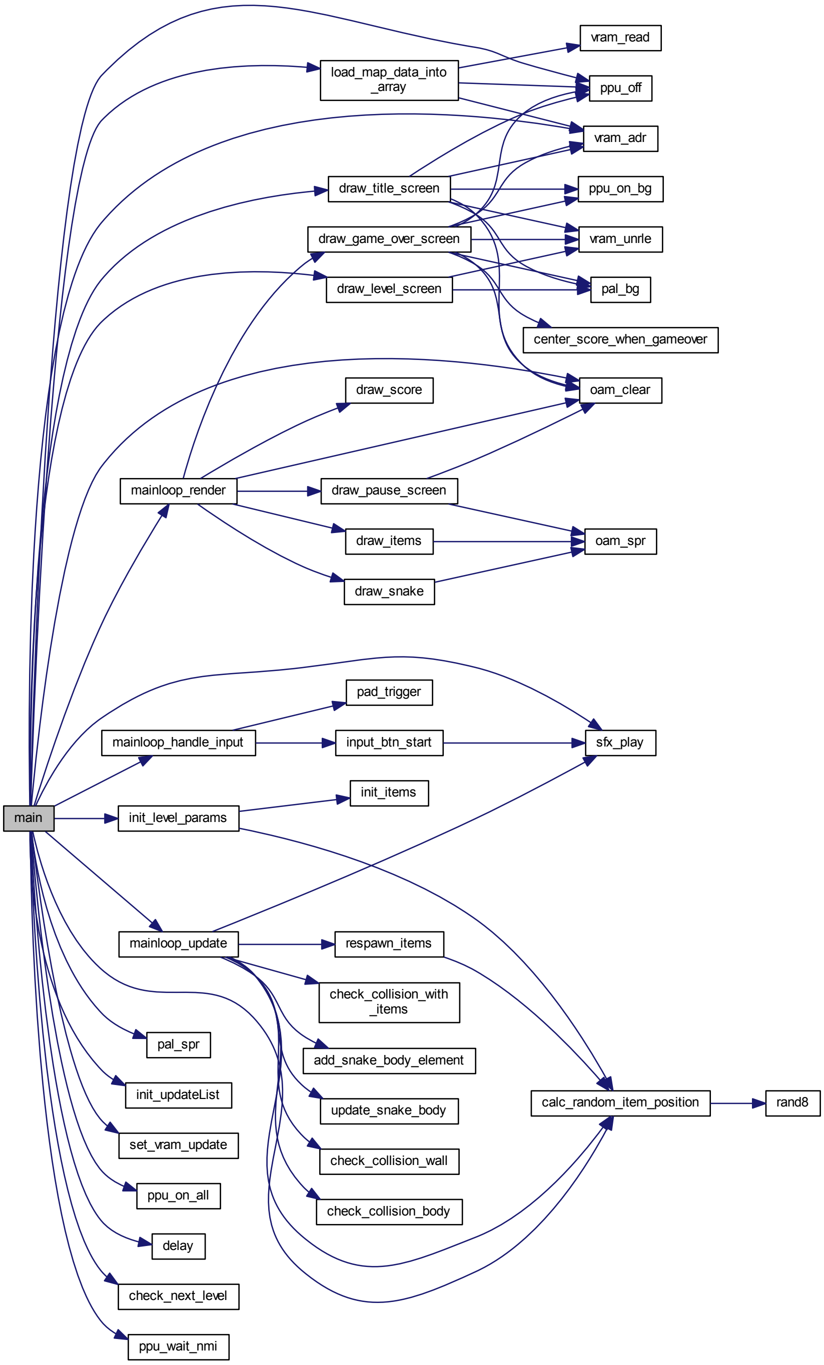
```
void main (
             void  )
```

Main game loop.

Author

Sebastian Dine

Here is the call graph for this function:



5.16 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/structures.h File Reference

This header file contains the definition of structures, created for the purpose of the game.

Data Structures

- struct snake_struct

  This structure contains all elements required to interact and display the snake.
- struct items_struct

  This structure contains all elements required to interact with and display items.

5.16.1 Detailed Description

This header file contains the definition of structures, created for the purpose of the game.

Author

Sebastian Dine

5.17 C:/Users/Administrator/Documents/GitHub/NES-Snake/src/update.c File Reference

This file contains all ingame logic functionalities and utility functionalities.

Functions

- void update_snake_body ()
- void add_snake_body_element ()
- unsigned char check_collision_wall (void)
- unsigned char check_collision_body (void)
- unsigned char check_collision_with_items (void)
- void respawn_items (void)
- unsigned char check_next_level (void)
- void mainloop_update (void)

5.17.1 Detailed Description

This file contains all ingame logic functionalities and utility functionalities.

Author

Sebastian Dine

5.17.2 Function Documentation

5.17.2.1 add_snake_body_element()

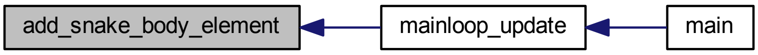
```
void add_snake_body_element ( )
```

This function adds a new pair of body element coordinates to global array 'body_coordinates'.

Author

Sebastian Dine

Here is the caller graph for this function:



5.17.2.2 check_collision_body()

```
unsigned char check_collision_body (
             void  )
```

Collision detecation of snakes' head-sprite with body-tiles.

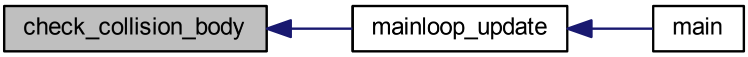
Returns

1 = collision with body element, 0 = no collision with body element

Author

Sebastian Dine

Here is the caller graph for this function:



5.17.2.3 check_collision_wall()

```
unsigned char check_collision_wall (
            void  )
```

Collision detection of snakes' head-sprite with wall-tiles.

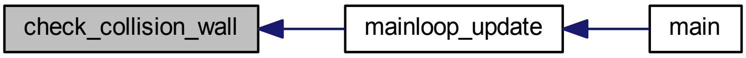
Returns

1 = collision with wall element, 0 = no collision with wall sprite

Author

Sebastian Dine

Here is the caller graph for this function:



5.17.2.4 check_collision_with_items()

```
unsigned char check_collision_with_items (
            void  )
```

Collision detection of snakes' head-sprite with an item-sprite.

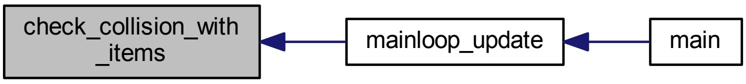
Returns

1 = collision with item sprite, 0 = no collision with item sprite

Author

Sebastian Dine

Here is the caller graph for this function:



5.17.2.5 check_next_level()

```
unsigned char check_next_level (
             void   )
```

Check, if the requirements for the next level are met.

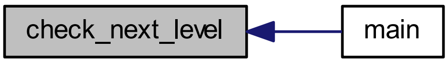
Returns

1 = next level is reached, 0 = next level is not reached

Author

Sebastian Dine

Here is the caller graph for this function:



5.17.2.6 mainloop_update()

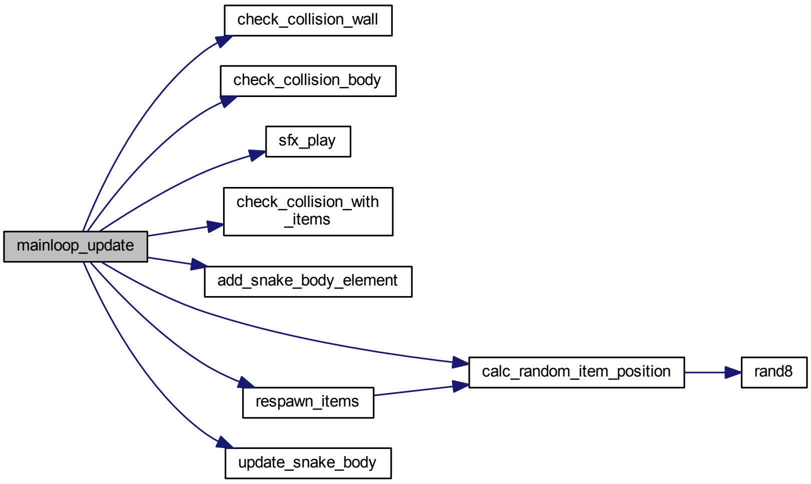
```
void mainloop_update (
             void   )
```

This function provides the coordination of all ingame logic routines, once per frame.

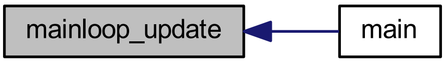
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.7 respawn_items()

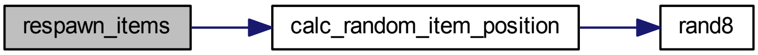
```
void respawn_items (
            void  )
```

This function counts down the frame rate for items until they respawn and calculates the new position in case of a respawn.

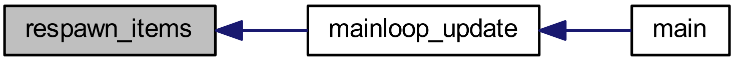
Author

Sebastian Dine

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.8 update_snake_body()

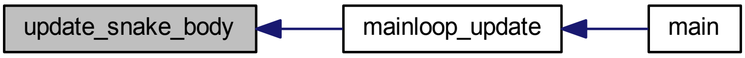
```
void update_snake_body ( )
```

This function updates the body coordinates of the snake in order to simulate its movement.

Author

Sebastian Dine

Here is the caller graph for this function:



Index