

Worms!

Generated by Doxygen 1.8.1.2

Sun May 11 2014 17:13:44

Contents

1	Worms!	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	cell Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	back	7
4.1.2.2	front	7
4.1.2.3	x	8
4.1.2.4	y	8
4.2	game_window Struct Reference	8
4.2.1	Field Documentation	8
4.2.1.1	cols	8
4.2.1.2	rows	8
4.2.1.3	window	8
4.3	gw Struct Reference	8
4.3.1	Field Documentation	9
4.3.1.1	initialized	9
4.3.1.2	old_cursor	9
4.3.1.3	window	9
4.3.1.4	ws	9
4.4	tge_parameters Struct Reference	9
4.4.1	Field Documentation	9
4.4.1.1	draw_function	9
4.4.1.2	input_function	9
4.4.1.3	setup_function	9

4.4.1.4	teardown_function	10
4.4.1.5	timer_interval	10
4.5	worm Struct Reference	10
4.5.1	Detailed Description	10
4.5.2	Field Documentation	10
4.5.2.1	head	10
4.5.2.2	last_direction	11
4.5.2.3	next_direction	11
4.5.2.4	tail	11
5	File Documentation	13
5.1	main.c File Reference	13
5.1.1	Detailed Description	13
5.1.2	Function Documentation	14
5.1.2.1	main	14
5.1.2.2	print_quit_message	14
5.2	tge.h File Reference	14
5.2.1	Detailed Description	15
5.3	tge_curses_routines.c File Reference	15
5.3.1	Detailed Description	16
5.3.2	Macro Definition Documentation	17
5.3.2.1	_POSIX_C_SOURCE	17
5.3.3	Function Documentation	17
5.3.3.1	tge_free_screen	17
5.3.3.2	tge_get_character	17
5.3.3.3	tge_initialize_screen	17
5.3.3.4	tge_main_window	17
5.3.3.5	tge_term_cols	17
5.3.3.6	tge_term_rows	17
5.3.4	Variable Documentation	18
5.3.4.1	game_window	18
5.4	tge_curses_routines.h File Reference	18
5.4.1	Detailed Description	19
5.4.2	Function Documentation	19
5.4.2.1	tge_free_screen	19
5.4.2.2	tge_get_character	19
5.4.2.3	tge_initialize_screen	19
5.4.2.4	tge_main_window	19
5.4.2.5	tge_term_cols	19
5.4.2.6	tge_term_rows	19

5.5	tge_game_state.c File Reference	20
5.5.1	Detailed Description	21
5.5.2	Enumeration Type Documentation	21
5.5.2.1	game_state	21
5.5.3	Function Documentation	21
5.5.3.1	tge_end_game	21
5.5.3.2	tge_end_status	21
5.5.3.3	tge_game_ended	21
5.5.3.4	tge_game_started	21
5.5.3.5	tge_needs_refresh	22
5.5.3.6	tge_set_needs_refresh	22
5.5.3.7	tge_start_game	22
5.5.4	Variable Documentation	22
5.5.4.1	game_state	22
5.5.4.2	refresh_flag	22
5.5.4.3	tge_end_status_var	22
5.6	tge_game_state.h File Reference	22
5.6.1	Detailed Description	23
5.6.2	Function Documentation	24
5.6.2.1	tge_end_game	24
5.6.2.2	tge_end_status	24
5.6.2.3	tge_game_ended	24
5.6.2.4	tge_game_started	24
5.6.2.5	tge_needs_refresh	24
5.6.2.6	tge_set_needs_refresh	25
5.6.2.7	tge_start_game	25
5.7	tge_main_game.c File Reference	25
5.7.1	Detailed Description	26
5.7.2	Macro Definition Documentation	26
5.7.2.1	_POSIX_C_SOURCE	26
5.7.3	Function Documentation	26
5.7.3.1	tge_begin_game	26
5.7.3.2	tge_game_loop	26
5.7.4	Variable Documentation	26
5.7.4.1	game_param	26
5.8	tge_main_game.h File Reference	26
5.8.1	Detailed Description	27
5.8.2	Function Documentation	27
5.8.2.1	tge_begin_game	27
5.8.2.2	tge_end_game	28

5.9	tge_signals.c File Reference	28
5.9.1	Detailed Description	29
5.9.2	Macro Definition Documentation	29
5.9.2.1	_POSIX_C_SOURCE	29
5.9.3	Function Documentation	29
5.9.3.1	tge_get_fractional_microseconds	29
5.9.3.2	tge_get_whole_seconds	29
5.9.3.3	tge_handler	29
5.9.3.4	tge_set_signal_handlers	30
5.9.3.5	tge_timer_start	30
5.9.3.6	tge_timer_stop	30
5.10	tge_signals.h File Reference	30
5.10.1	Detailed Description	30
5.10.2	Function Documentation	31
5.10.2.1	tge_set_signal_handlers	31
5.10.2.2	tge_timer_start	31
5.10.2.3	tge_timer_stop	31
5.11	worms.dox File Reference	31
5.12	worms.h File Reference	31
5.12.1	Detailed Description	32
5.13	worms_food.c File Reference	32
5.13.1	Detailed Description	33
5.13.2	Function Documentation	33
5.13.2.1	worms_draw_food	33
5.13.2.2	worms_food_here	33
5.13.2.3	worms_get_food_eaten	33
5.13.2.4	worms_place_new_food	33
5.13.3	Variable Documentation	33
5.13.3.1	food_eaten	34
5.13.3.2	food_x	34
5.13.3.3	food_y	34
5.14	worms_food.h File Reference	34
5.14.1	Detailed Description	34
5.14.2	Function Documentation	35
5.14.2.1	worms_draw_food	35
5.14.2.2	worms_food_here	35
5.14.2.3	worms_get_food_eaten	35
5.14.2.4	worms_place_new_food	35
5.15	worms_game.c File Reference	35
5.15.1	Detailed Description	36

5.15.2	Function Documentation	36
5.15.2.1	worms_draw_screen	36
5.15.2.2	worms_game_setup	36
5.15.2.3	worms_game_teardown	36
5.15.2.4	worms_process_input	36
5.16	worms_game.h File Reference	37
5.16.1	Detailed Description	37
5.16.2	Enumeration Type Documentation	37
5.16.2.1	worms_exit_status	37
5.16.3	Function Documentation	38
5.16.3.1	worms_draw_screen	38
5.16.3.2	worms_game_setup	38
5.16.3.3	worms_game_teardown	38
5.16.3.4	worms_process_input	38
5.17	worms_screen.c File Reference	38
5.17.1	Detailed Description	39
5.17.2	Function Documentation	40
5.17.2.1	worms_coords_in_game_arena	40
5.17.2.2	worms_draw_arena_border	40
5.17.2.3	worms_draw_info_window	40
5.17.2.4	worms_draw_sidebar	40
5.17.2.5	worms_draw_title_window	40
5.17.2.6	worms_game_area_destroy	40
5.17.2.7	worms_game_area_init	40
5.17.2.8	worms_game_arena_cols	40
5.17.2.9	worms_game_arena_rows	41
5.17.2.10	worms_get_arena_character	41
5.17.2.11	worms_refresh_game_area	41
5.17.2.12	worms_write_arena_character	41
5.17.3	Variable Documentation	41
5.17.3.1	arena_window	41
5.17.3.2	info_window	41
5.17.3.3	sidebar_cols	41
5.17.3.4	title_rows	42
5.17.3.5	title_window	42
5.18	worms_screen.h File Reference	42
5.18.1	Detailed Description	43
5.18.2	Enumeration Type Documentation	43
5.18.2.1	worms_cell_character	43
5.18.3	Function Documentation	43

5.18.3.1	worms_coords_in_game_arena	43
5.18.3.2	worms_draw_arena_border	44
5.18.3.3	worms_draw_sidebar	44
5.18.3.4	worms_game_area_destroy	44
5.18.3.5	worms_game_area_init	44
5.18.3.6	worms_game_arena_cols	44
5.18.3.7	worms_game_arena_rows	44
5.18.3.8	worms_get_arena_character	44
5.18.3.9	worms_refresh_game_area	45
5.18.3.10	worms_write_arena_character	45
5.19	worms_time.c File Reference	45
5.19.1	Detailed Description	45
5.19.2	Function Documentation	46
5.19.2.1	worms_game_time	46
5.19.2.2	worms_game_time_string	46
5.19.2.3	worms_time_init	46
5.19.3	Variable Documentation	46
5.19.3.1	start_time	46
5.20	worms_time.h File Reference	46
5.20.1	Detailed Description	47
5.20.2	Function Documentation	47
5.20.2.1	worms_game_time	47
5.20.2.2	worms_game_time_string	47
5.20.2.3	worms_time_init	48
5.21	worms_worm.c File Reference	48
5.21.1	Detailed Description	49
5.21.2	Function Documentation	49
5.21.2.1	worm_add_cell_head	49
5.21.2.2	worm_cell_here	49
5.21.2.3	worm_clear	49
5.21.2.4	worm_delete_cell_head	50
5.21.2.5	worm_delete_cell_tail	50
5.21.2.6	worm_destroy	50
5.21.2.7	worm_draw	50
5.21.2.8	worm_init	50
5.21.2.9	worm_move	50
5.21.2.10	worm_move_and_draw	50
5.21.2.11	worm_set_direction	50
5.21.3	Variable Documentation	51
5.21.3.1	worm	51

5.21.3.2	WORM_START_LENGTH	51
5.22	worms_worm.h File Reference	51
5.22.1	Detailed Description	52
5.22.2	Enumeration Type Documentation	52
5.22.2.1	worm_direction	52
5.22.3	Function Documentation	52
5.22.3.1	worm_destroy	52
5.22.3.2	worm_init	52
5.22.3.3	worm_move_and_draw	52
5.22.3.4	worm_set_direction	53

Chapter 1

Worms!

Worms! is an ncurses based worms game.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

cell	7
game_window	Structure to hold details of game windows	8
gw	Structure for main game window	8
tge_parameters	Structure for containing game parameters	9
worm	10

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

main.c	Main() function for worms game	13
tge.h	Aggregated header for curses timer game engine functions	14
tge_curses_routines.c	Implementation of TGE general curses functions	15
tge_curses_routines.h	Interface to TGE general curses functions	18
tge_game_state.c	Implementation of TGE game state functions	20
tge_game_state.h	Interface to TGE game state functions	22
tge_main_game.c	Implementation of TGE main game functions	25
tge_main_game.h	Interface to curses timer game engine main game functions	26
tge_signals.c	Implementation of TGE signals functions	28
tge_signals.h	Interface to TGE signals functions	30
worms.h	Aggregated header for worms game functions	31
worms_food.c	Implementation of worm food functions	32
worms_food.h	Interface to worm food functions	34
worms_game.c	Implementation of TGE game engine callback functions	35
worms_game.h	Interface to TGE game engine callback functions	37
worms_screen.c	Implementation of worms game screen functions	38
worms_screen.h	Interface to worms game screen functions	42
worms_time.c	Implementation of worms game duration timer functions	45
worms_time.h	Interface to worms game duration timer functions	46

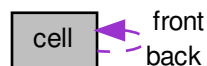
worms_worm.c	
Implementation of worms game worm functions	48
worms_worm.h	
Interface to worms game worm functions	51

Chapter 4

Data Structure Documentation

4.1 cell Struct Reference

Collaboration diagram for cell:



Data Fields

- int `x`
- int `y`
- struct `cell` * `front`
- struct `cell` * `back`

4.1.1 Detailed Description

Structure to hold an individual cell of the worm's body

4.1.2 Field Documentation

4.1.2.1 struct `cell`* `cell::back`

Pointer to next cell towards tail

4.1.2.2 struct `cell`* `cell::front`

Pointer to next cell towards head

4.1.2.3 int cell::x

X coordinate of the cell

4.1.2.4 int cell::y

Y coordinate of the cell

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

- [worms_worm.c](#)

4.2 game_window Struct Reference

Structure to hold details of game windows.

Data Fields

- WINDOW * [window](#)
- int [rows](#)
- int [cols](#)

4.2.1 Field Documentation

4.2.1.1 int game_window::cols

Number of columns in the window

4.2.1.2 int game_window::rows

Number of rows in the window

4.2.1.3 WINDOW* game_window::window

Pointer to curses WINDOW structure

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

- [worms_screen.c](#)

4.3 gw Struct Reference

Structure for main game window.

Data Fields

- WINDOW * [window](#)
- struct winsize [ws](#)
- int [old_cursor](#)
- bool [initialized](#)

4.3.1 Field Documentation

4.3.1.1 bool gw::initialized

true if initialized, false otherwise

4.3.1.2 int gw::old_cursor

To store the old cursor

4.3.1.3 WINDOW* gw::window

Pointer to main curses window

4.3.1.4 struct winsize gw::ws

Contains dimensions of terminal

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

- [tge_curses_routines.c](#)

4.4 tge_parameters Struct Reference

Structure for containing game parameters.

```
#include <tge_main_game.h>
```

Data Fields

- void(* [setup_function](#))(void)
- void(* [draw_function](#))(void)
- void(* [input_function](#))(int)
- void(* [teardown_function](#))(int)
- double [timer_interval](#)

4.4.1 Field Documentation

4.4.1.1 void(* tge_parameters::draw_function)(void)

Draw function

4.4.1.2 void(* tge_parameters::input_function)(int)

Input handling function

4.4.1.3 void(* tge_parameters::setup_function)(void)

Setup/initialization function

4.4.1.4 void(* tge_parameters::teardown_function)(int)

Cleanup function

4.4.1.5 double tge_parameters::timer_interval

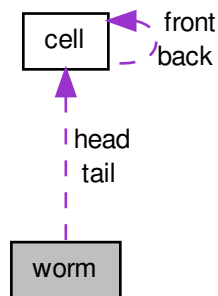
Timer interval, in seconds

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

- [tge_main_game.h](#)

4.5 worm Struct Reference

Collaboration diagram for worm:



Data Fields

- struct [cell](#) * [head](#)
- struct [cell](#) * [tail](#)
- enum [worm_direction](#) [last_direction](#)
- enum [worm_direction](#) [next_direction](#)

4.5.1 Detailed Description

Structure to hold the worm

4.5.2 Field Documentation

4.5.2.1 struct [cell](#)* worm::head

Pointer to head cell

4.5.2.2 enum worm_direction worm::last_direction

Next direction to move

4.5.2.3 enum worm_direction worm::next_direction

Last direction moved

4.5.2.4 struct cell* worm::tail

Pointer to tail cell

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

- [worms_worm.c](#)

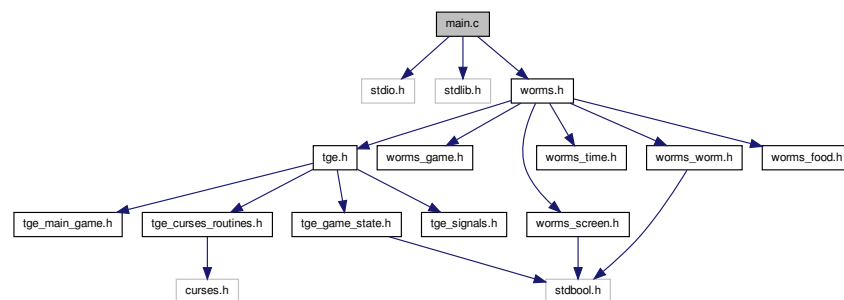
Chapter 5

File Documentation

5.1 main.c File Reference

[main\(\)](#) function for worms game.

```
#include <stdio.h>
#include <stdlib.h>
#include "worms.h"
Include dependency graph for main.c:
```



Functions

- void [print_quit_message](#) (const int end_status)
Prints a quit message.
- int [main](#) (void)
[main\(\)](#) function.

5.1.1 Detailed Description

[main\(\)](#) function for worms game.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.1.2 Function Documentation

5.1.2.1 `int main (void)`

`main()` function.

Returns

The exit status of the program.

5.1.2.2 `void print_quit_message (const int end_status)`

Prints a quit message.

Parameters

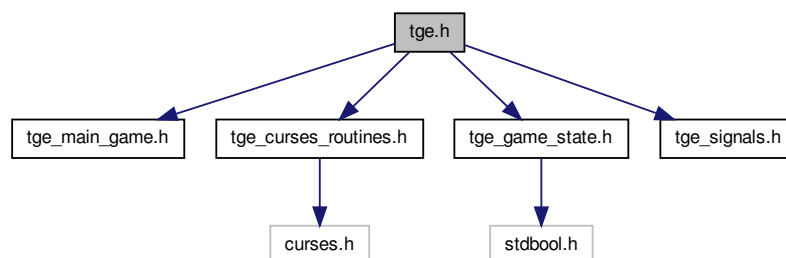
<i>end_status</i>	The exit status of the game.
-------------------	------------------------------

5.2 tge.h File Reference

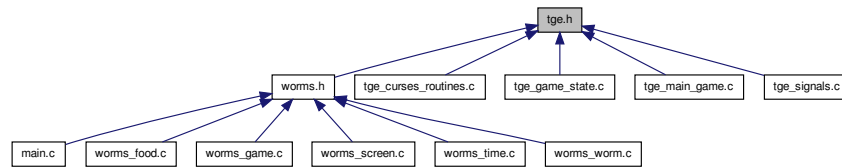
Aggregated header for curses timer game engine functions.

```
#include "tge_main_game.h"
#include "tge_curses_routines.h"
#include "tge_game_state.h"
#include "tge_signals.h"
```

Include dependency graph for tge.h:



This graph shows which files directly or indirectly include this file:



5.2.1 Detailed Description

Aggregated header for curses timer game engine functions.

A "timer game" is here defined as a game with two properties:

- the game and display regularly updates at a specified time interval; and
- the game can receive keyboard input at any time.

The rationale for providing this framework is twofold. Firstly, to encapsulate some of the overhead of setting up the game, including initializing curses, setting up signal handlers, and controlling the overall flow of the game. Secondly, to solve the programming problem of asynchronously updating the game while waiting on user input.

An initial naive implementation may be to implement the game timer using `SIGALRM`, and update the screen in the signal handler, while simply blocking on user input in the main thread. However, the curses functions are not re-entrant, and neither are many standard library functions, so this approach is not reliable. Similar objections would apply to a threads-based approach where one thread blocks on input, and a second thread updates the screen.

A second naive implementation would be to simply poll for input without blocking, which would be an unnecessary waste of processor time.

The solution here is to wait for input with `select()` to avoid blocking on any input. The game timer is implemented with `SIGALRM`, and the signal handler simply sets an "updated needed" variable and has the automatic side-effect of interrupting `select()` (note: portable curses programs cannot make any assumptions about whether handled signals will interrupt any curses input functions, so `select()` is necessary).

The library implements a game loop which begins by checking if a screen update is necessary. If it is, the screen is updated, and the loop re-entered. If it is not, the loop waits on `select()`. If no input is entered, `select()` will interrupt on handling `SIGALRM` and continue to the next iteration of the loop, and the screen will be updated again. If input is entered, `select()` will return and the input can be obtained without blocking.

Author

Paul Griffiths

Copyright

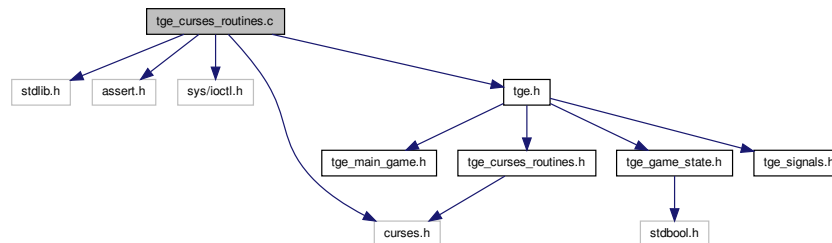
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.3 tge_curses_routines.c File Reference

Implementation of TGE general curses functions.

```
#include <stdlib.h>
#include <assert.h>
#include <sys/ioctl.h>
#include <curses.h>
#include "tge.h"
```

Include dependency graph for `tge_curses_routines.c`:



Data Structures

- struct `gw`
Structure for main game window.

Macros

- `#define _POSIX_C_SOURCE 200809L`

Functions

- void `tge_initialize_screen` (void)
- void `tge_free_screen` (void)
- WINDOW * `tge_main_window` (void)
Returns a pointer to the main curses window.
- int `tge_get_character` (void)
Gets a character input by the player.
- int `tge_term_rows` (void)
Returns the number of rows in the terminal.
- int `tge_term_cols` (void)
Returns the number of columns in the terminal.

Variables

- static struct `gw game_window` = {NULL, {0, 0, 0, 0}, 0, false}

5.3.1 Detailed Description

Implementation of TGE general curses functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.3.2 Macro Definition Documentation

5.3.2.1 `#define _POSIX_C_SOURCE 200809L`

POSIX feature test macro

5.3.3 Function Documentation

5.3.3.1 `void tge_free_screen (void)`

Frees and destroys the game screen.

5.3.3.2 `int tge_get_character (void)`

Gets a character input by the player. This function will not block if no input is ready.

Returns

The character input by the player, or -1 if no character was ready.

5.3.3.3 `void tge_initialize_screen (void)`

Initializes the game screen.

5.3.3.4 `WINDOW* tge_main_window (void)`

Returns a pointer to the main curses window.

Returns

A pointer to the main curses window.

5.3.3.5 `int tge_term_cols (void)`

Returns the number of columns in the terminal.

Returns

The number of columns in the terminal.

5.3.3.6 `int tge_term_rows (void)`

Returns the number of rows in the terminal.

Returns

The number of rows in the terminal.

5.3.4 Variable Documentation

5.3.4.1 `struct gw game_window = {NULL, {0, 0, 0, 0}, 0, false}` `[static]`

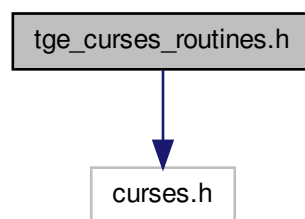
File scope variable to hold main game window

5.4 tge_curses_routines.h File Reference

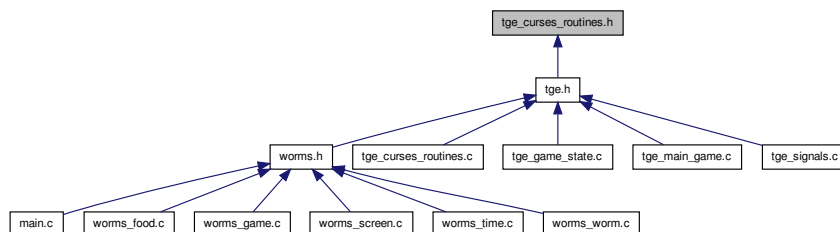
Interface to TGE general curses functions.

```
#include <curses.h>
```

Include dependency graph for tge_curses_routines.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [tge_initialize_screen](#) (void)
- void [tge_free_screen](#) (void)
- WINDOW * [tge_main_window](#) (void)
Returns a pointer to the main curses window.
- int [tge_get_character](#) (void)
Gets a character input by the player.
- int [tge_term_rows](#) (void)
Returns the number of rows in the terminal.
- int [tge_term_cols](#) (void)
Returns the number of columns in the terminal.

5.4.1 Detailed Description

Interface to TGE general curses functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.4.2 Function Documentation

5.4.2.1 void tge_free_screen (void)

Frees and destroys the game screen.

5.4.2.2 int tge_get_character (void)

Gets a character input by the player. This function will not block if no input is ready.

Returns

The character input by the player, or -1 if no character was ready.

5.4.2.3 void tge_initialize_screen (void)

Initializes the game screen.

5.4.2.4 WINDOW* tge_main_window (void)

Returns a pointer to the main curses window.

Returns

A pointer to the main curses window.

5.4.2.5 int tge_term_cols (void)

Returns the number of columns in the terminal.

Returns

The number of columns in the terminal.

5.4.2.6 int tge_term_rows (void)

Returns the number of rows in the terminal.

Returns

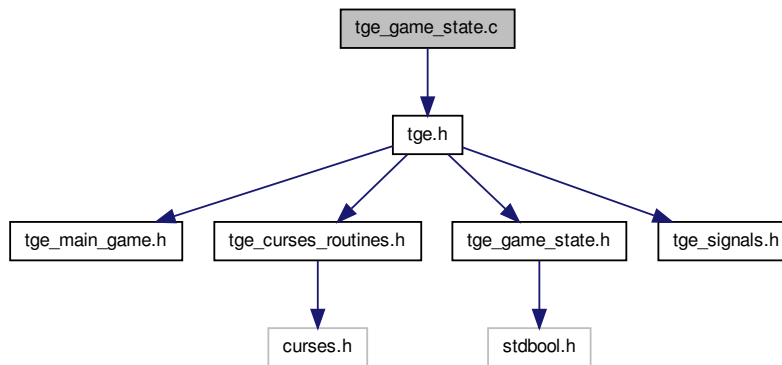
The number of rows in the terminal.

5.5 tge_game_state.c File Reference

Implementation of TGE game state functions.

```
#include "tge.h"
```

Include dependency graph for tge_game_state.c:



Enumerations

- enum [game_state](#) { [TGE_GAME_STATE_NOTSTARTED](#), [TGE_GAME_STATE_RUNNING](#), [TGE_GAME_STATE_ENDED](#) }

Enumeration constants for game state.

Functions

- void [tge_start_game](#) (void)
- bool [tge_game_started](#) (void)
Tests if the timer game has started.
- void [tge_end_game](#) (const int status)
Ends the timer game.
- bool [tge_game_ended](#) (void)
Tests if the timer game has ended.
- int [tge_end_status](#) (void)
Returns the timer game exit status.
- void [tge_set_needs_refresh](#) (const bool status)
Notifies the library that the game needs updating.
- bool [tge_needs_refresh](#) (void)
Checks if the game needs updating.

Variables

- static int [tge_end_status_var](#) = 0
- static enum [game_state](#) [game_state](#) = [TGE_GAME_STATE_NOTSTARTED](#)
- static bool [refresh_flag](#) = true

5.5.1 Detailed Description

Implementation of TGE game state functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.5.2 Enumeration Type Documentation

5.5.2.1 enum game_state

Enumerator:

TGE_GAME_STATE_NOTSTARTED Game has not started

TGE_GAME_STATE_RUNNING Game has started

TGE_GAME_STATE_ENDED Game has ended

5.5.3 Function Documentation

5.5.3.1 void tge_end_game (const int *status*)

Ends the timer game.

Parameters

<i>status</i>	The exit status of the game.
---------------	------------------------------

5.5.3.2 int tge_end_status (void)

Returns the timer game exit status. The meaning of this value is defined by the application using the TGE library.

Returns

The timer game exit status.

5.5.3.3 bool tge_game_ended (void)

Tests if the timer game has ended.

Returns

`true` if the timer game has ended, `false` otherwise.

5.5.3.4 bool tge_game_started (void)

Tests if the timer game has started.

Returns

`true` if the timer game has started, `false` otherwise.

5.5.3.5 bool tge_needs_refresh (void)

Checks if the game needs updating. This function returns the state set by `tge_set_needs_refresh()`.

Returns

`true` is the game needs updating, `false` otherwise.

5.5.3.6 void tge_set_needs_refresh (const bool status)

Notifies the library that the game needs updating. This is typically called by the `SIGALRM` signal handler in response to the game timer, but could be called by the application.

Parameters

<i>status</i>	<code>true</code> to turn on the "needs updating" flag, <code>false</code> to turn it off.
---------------	--

5.5.3.7 void tge_start_game (void)

Starts the timer game.

5.5.4 Variable Documentation**5.5.4.1 enum game_state game_state = TGE_GAME_STATE_NOTSTARTED [static]**

File scope variable for current game state

5.5.4.2 bool refresh_flag = true [static]

File scope variable for "needs updating" flag

5.5.4.3 int tge_end_status_var = 0 [static]

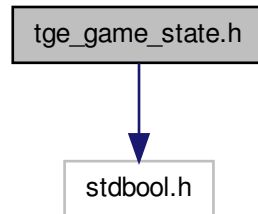
File scope variable for current exit status

5.6 tge_game_state.h File Reference

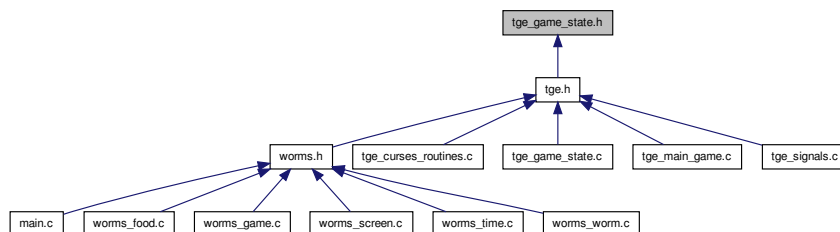
Interface to TGE game state functions.


```
#include <stdbool.h>
```

Include dependency graph for tge_game_state.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [tge_start_game](#) (void)
- bool [tge_game_started](#) (void)
Tests if the timer game has started.
- void [tge_end_game](#) (const int status)
Ends the timer game.
- bool [tge_game_ended](#) (void)
Tests if the timer game has ended.
- int [tge_end_status](#) (void)
Returns the timer game exit status.
- void [tge_set_needs_refresh](#) (const bool status)
Notifies the library that the game needs updating.
- bool [tge_needs_refresh](#) (void)
Checks if the game needs updating.

5.6.1 Detailed Description

Interface to TGE game state functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.6.2 Function Documentation**5.6.2.1 void tge_end_game (const int *status*)**

Ends the timer game.

Parameters

<i>status</i>	The exit status of the game.
---------------	------------------------------

5.6.2.2 int tge_end_status (void)

Returns the timer game exit status. The meaning of this value is defined by the application using the TGE library.

Returns

The timer game exit status.

5.6.2.3 bool tge_game_ended (void)

Tests if the timer game has ended.

Returns

`true` if the timer game has ended, `false` otherwise.

5.6.2.4 bool tge_game_started (void)

Tests if the timer game has started.

Returns

`true` if the timer game has started, `false` otherwise.

5.6.2.5 bool tge_needs_refresh (void)

Checks if the game needs updating. This function returns the state set by `tge_set_needs_refresh()`.

Returns

`true` is the game needs updating, `false` otherwise.

5.6.2.6 void tge_set_needs_refresh (const bool *status*)

Notifies the library that the game needs updating. This is typically called by the `SIGALRM` signal handler in response to the game timer, but could be called by the application.

Parameters

<i>status</i>	true to turn on the "needs updating" flag, false to turn it off.
---------------	--

5.6.2.7 void tge_start_game (void)

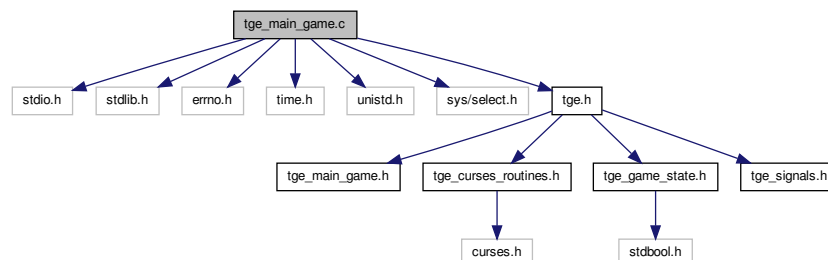
Starts the timer game.

5.7 tge_main_game.c File Reference

Implementation of TGE main game functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <unistd.h>
#include <sys/select.h>
#include "tge.h"
```

Include dependency graph for tge_main_game.c:



Macros

- `#define _POSIX_C_SOURCE 200809L`

Functions

- static void `tge_game_loop` (void)
- int `tge_begin_game` (const struct `tge_parameters` *parameters)
Begins the timer game.

Variables

- static struct `tge_parameters` `game_param`

5.7.1 Detailed Description

Implementation of TGE main game functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.7.2 Macro Definition Documentation

5.7.2.1 `#define _POSIX_C_SOURCE 200809L`

POSIX feature test macro

5.7.3 Function Documentation

5.7.3.1 `int tge_begin_game (const struct tge_parameters * parameters)`

Begins the timer game.

Parameters

<i>parameters</i>	A pointer to a struct tge_parameters object containing the desired game parameters.
-------------------	---

Returns

The exit status of the game.

5.7.3.2 `static void tge_game_loop (void) [static]`

Main game loop function.

5.7.4 Variable Documentation

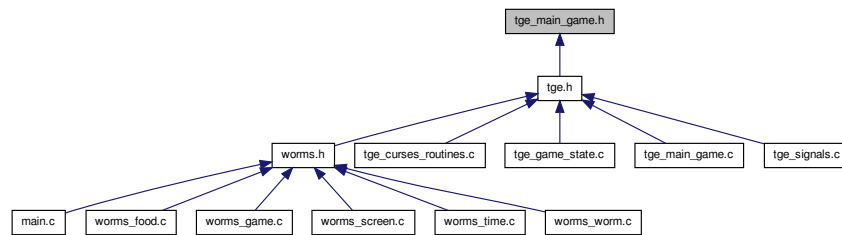
5.7.4.1 `struct tge_parameters game_param [static]`

File scope variable for game parameters

5.8 `tge_main_game.h` File Reference

Interface to curses timer game engine main game functions.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [tge_parameters](#)
Structure for containing game parameters.

Functions

- int [tge_begin_game](#) (const struct [tge_parameters](#) *parameters)
Begins the timer game.
- void [tge_end_game](#) (const int status)
Ends the timer game.

5.8.1 Detailed Description

Interface to curses timer game engine main game functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.8.2 Function Documentation

5.8.2.1 int tge_begin_game (const struct tge_parameters * parameters)

Begins the timer game.

Parameters

<i>parameters</i>	A pointer to a struct tge_parameters object containing the desired game parameters.
-------------------	---

Returns

The exit status of the game.

5.8.2.2 void tge_end_game (const int status)

Ends the timer game.

Parameters

<i>status</i>	The exit status for the game.
---------------	-------------------------------

Ends the timer game.

Parameters

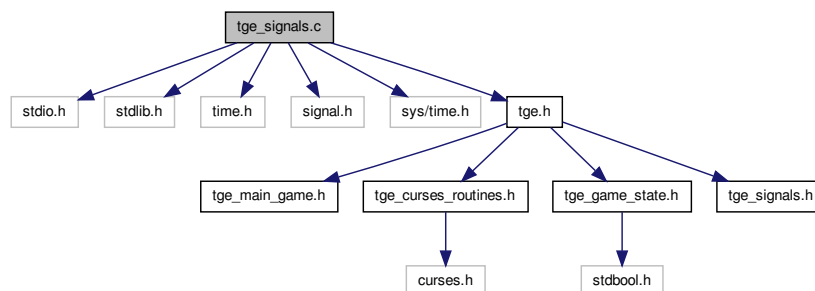
<i>status</i>	The exit status of the game.
---------------	------------------------------

5.9 tge_signals.c File Reference

Implementation of TGE signals functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <signal.h>
#include <sys/time.h>
#include "tge.h"
```

Include dependency graph for tge_signals.c:



Macros

- `#define _POSIX_C_SOURCE 200809L`

Functions

- static long [tge_get_whole_seconds](#) (const double seconds)
Returns the whole seconds part of a seconds value.
- static long [tge_get_fractional_microseconds](#) (const double seconds)
Returns the fractional microseconds part of a seconds value.
- static void [tge_handler](#) (int signum)
Generic signal handler.
- void [tge_set_signal_handlers](#) (void)
- void [tge_timer_start](#) (const double start, const double interval)

Starts the game timer.

- void `tge_timer_stop` (void)

5.9.1 Detailed Description

Implementation of TGE signals functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.9.2 Macro Definition Documentation

5.9.2.1 `#define _POSIX_C_SOURCE 200809L`

POSIX feature test macro

5.9.3 Function Documentation

5.9.3.1 `static long tge_get_fractional_microseconds (const double seconds) [static]`

Returns the whole microseconds part of a seconds value.

Parameters

<i>seconds</i>	The seconds representation.
----------------	-----------------------------

Returns

The fractional microseconds part of the seconds value.

5.9.3.2 `static long tge_get_whole_seconds (const double seconds) [static]`

Returns the whole seconds part of a seconds value.

Parameters

<i>seconds</i>	The seconds value.
----------------	--------------------

Returns

The whole seconds part of the seconds value.

5.9.3.3 `static void tge_handler (int signum) [static]`

Generic signal handler.

Parameters

<i>signum</i>	The signal number.
---------------	--------------------

5.9.3.4 void tge_set_signal_handlers (void)

Registers the signal handlers.

5.9.3.5 void tge_timer_start (const double start, const double interval)

Starts the game timer.

Parameters

<i>start</i>	The time until the first alarm, in seconds.
<i>interval</i>	The time interval between subsequent alarms, in seconds.

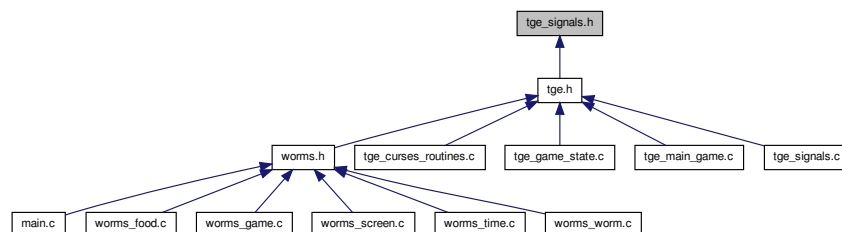
5.9.3.6 void tge_timer_stop (void)

Stops the game timer.

5.10 tge_signals.h File Reference

Interface to TGE signals functions.

This graph shows which files directly or indirectly include this file:



Functions

- void [tge_set_signal_handlers](#) (void)
- void [tge_timer_start](#) (const double start, const double interval)
Starts the game timer.
- void [tge_timer_stop](#) (void)

5.10.1 Detailed Description

Interface to TGE signals functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.10.2 Function Documentation**5.10.2.1 void tge_set_signal_handlers (void)**

Registers the signal handlers.

5.10.2.2 void tge_timer_start (const double *start*, const double *interval*)

Starts the game timer.

Parameters

<i>start</i>	The time until the first alarm, in seconds.
<i>interval</i>	The time interval between subsequent alarms, in seconds.

5.10.2.3 void tge_timer_stop (void)

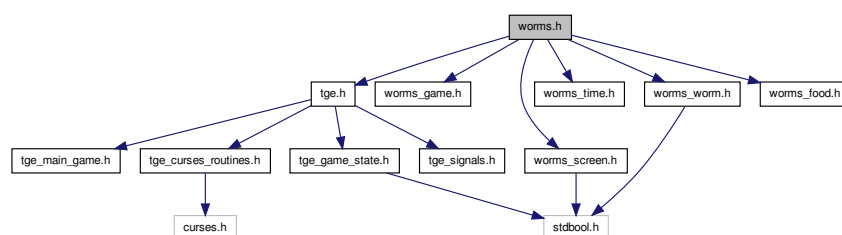
Stops the game timer.

5.11 worms.dox File Reference**5.12 worms.h File Reference**

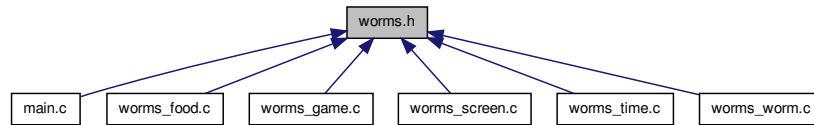
Aggregated header for worms game functions.

```
#include "tge.h"
#include "worms_game.h"
#include "worms_screen.h"
#include "worms_time.h"
#include "worms_worm.h"
#include "worms_food.h"
```

Include dependency graph for worms.h:



This graph shows which files directly or indirectly include this file:



5.12.1 Detailed Description

Aggregated header for worms game functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

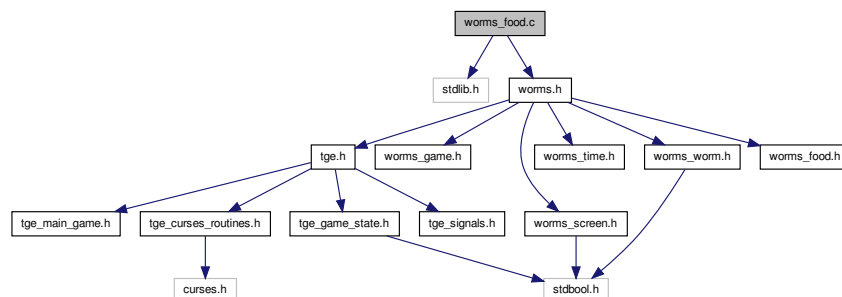
5.13 worms_food.c File Reference

Implementation of worm food functions.

```
#include <stdlib.h>
```

```
#include "worms.h"
```

Include dependency graph for worms_food.c:



Functions

- void [worms_place_new_food](#) (void)
- void [worms_draw_food](#) (void)
- bool [worms_food_here](#) (const int x, const int y)
Tests if a piece of food is at the specified coordinates.
- int [worms_get_food_eaten](#) (void)
Returns the total number of pieces of food eaten.

Variables

- static int `food_x` = 0
- static int `food_y` = 0
- static int `food_eaten` = -1

5.13.1 Detailed Description

Implementation of worm food functions for curses worms game.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.13.2 Function Documentation

5.13.2.1 void worms_draw_food (void)

Draws the current piece of food.

5.13.2.2 bool worms_food_here (const int x, const int y)

Tests if a piece of food is at the specified coordinates.

Parameters

<code>x</code>	The specified x coordinate.
<code>y</code>	The specified y coordinate.

Returns

`true` if a piece of food is at the specified coordinates, `false` otherwise.

5.13.2.3 int worms_get_food_eaten (void)

Returns the total number of pieces of food eaten since the start of the game.

Returns

The total number of pieces of food eaten since the start of the game.

5.13.2.4 void worms_place_new_food (void)

Places a new piece of food at a random location.

5.13.3 Variable Documentation

5.13.3.1 `int food_eaten = -1` `[static]`

File scope variable for total food eaten since start of game

5.13.3.2 `int food_x = 0` `[static]`

File scope variable for x coordinate of current food piece

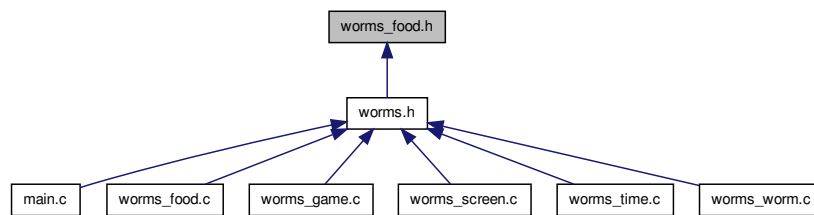
5.13.3.3 `int food_y = 0` `[static]`

File scope variable for y coordinate of current food piece

5.14 worms_food.h File Reference

Interface to worm food functions.

This graph shows which files directly or indirectly include this file:



Functions

- void `worms_place_new_food` (void)
- void `worms_draw_food` (void)
- bool `worms_food_here` (const int x, const int y)
Tests if a piece of food is at the specified coordinates.
- int `worms_get_food_eaten` (void)
Returns the total number of pieces of food eaten.

5.14.1 Detailed Description

Interface to worm food functions for curses worms game.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.14.2 Function Documentation

5.14.2.1 void worms_draw_food (void)

Draws the current piece of food.

5.14.2.2 bool worms_food_here (const int x, const int y)

Tests if a piece of food is at the specified coordinates.

Parameters

x	The specified x coordinate.
y	The specified y coordinate.

Returns

`true` if a piece of food is at the specified coordinates, `false` otherwise.

5.14.2.3 int worms_get_food_eaten (void)

Returns the total number of pieces of food eaten since the start of the game.

Returns

The total number of pieces of food eaten since the start of the game.

5.14.2.4 void worms_place_new_food (void)

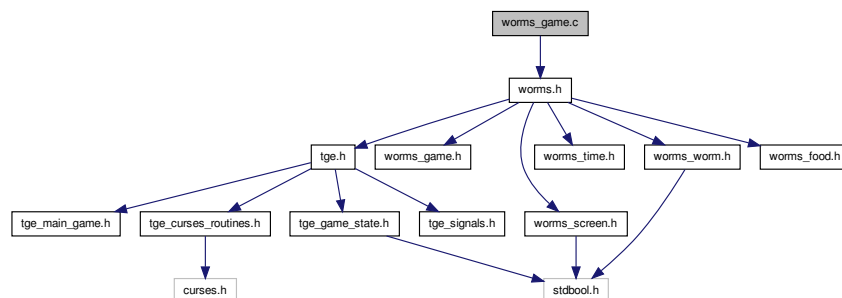
Places a new piece of food at a random location.

5.15 worms_game.c File Reference

Implementation of TGE game engine callback functions.

```
#include "worms.h"
```

Include dependency graph for worms_game.c:



Functions

- void `worms_game_setup` (void)
Initializes the game.
- void `worms_game_teardown` (const int end_status)
Cleans up after the game ends.
- void `worms_draw_screen` (void)
- void `worms_process_input` (const int ch)
Processes keyboard input.

5.15.1 Detailed Description

Implementation of TGE game engine callback functions for curses worms game.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.15.2 Function Documentation

5.15.2.1 void worms_draw_screen (void)

Draws the game screen at each timer interval.

5.15.2.2 void worms_game_setup (void)

Initializes the game area, worm, worm food, and game time.

5.15.2.3 void worms_game_teardown (const int end_status)

Destroys the worm and the game area.

Parameters

<i>end_status</i>	Status code for end-of-game reason.
-------------------	-------------------------------------

5.15.2.4 void worms_process_input (const int ch)

Processes keyboard input to move the worm or quit the game.

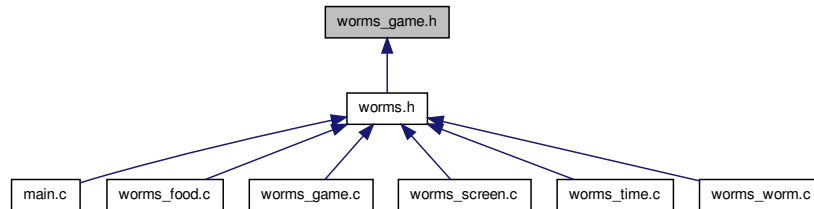
Parameters

<i>ch</i>	Character code representing the key pressed.
-----------	--

5.16 worms_game.h File Reference

Interface to TGE game engine callback functions.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum `worms_exit_status` { `WORMS_EXIT_NORMAL`, `WORMS_EXIT_HITWALL`, `WORMS_EXIT_HITSELF` }

Functions

- void `worms_game_setup` (void)
Initializes the game.
- void `worms_game_teardown` (const int end_status)
Cleans up after the game ends.
- void `worms_draw_screen` (void)
- void `worms_process_input` (const int ch)
Processes keyboard input.

5.16.1 Detailed Description

Interface to TGE game engine callback functions, along with general game data structures and constants.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.16.2 Enumeration Type Documentation

5.16.2.1 enum worms_exit_status

Enumeration constants for game exit status.

Enumerator:

`WORMS_EXIT_NORMAL` Player chose to quit

WORMS_EXIT_HITWALL Worm ran into the arena wall

WORMS_EXIT_HITSELF Worm ran into its own body

5.16.3 Function Documentation

5.16.3.1 void worms_draw_screen (void)

Draws the game screen at each timer interval.

5.16.3.2 void worms_game_setup (void)

Initializes the game area, worm, worm food, and game time.

5.16.3.3 void worms_game_teardown (const int *end_status*)

Destroys the worm and the game area.

Parameters

<i>end_status</i>	Status code for end-of-game reason.
-------------------	-------------------------------------

5.16.3.4 void worms_process_input (const int *ch*)

Processes keyboard input to move the worm or quit the game.

Parameters

<i>ch</i>	Character code representing the key pressed.
-----------	--

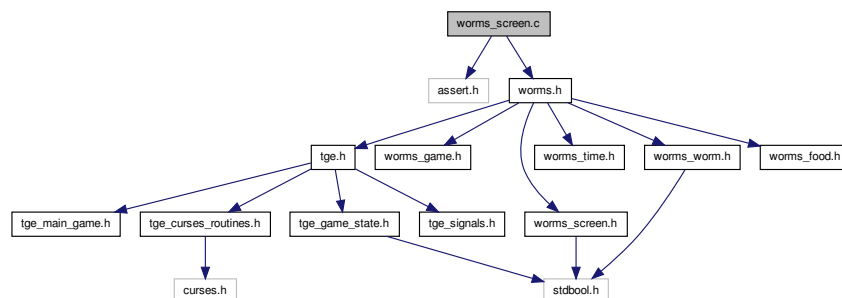
5.17 worms_screen.c File Reference

Implementation of worms game screen functions.

```
#include <assert.h>
```

```
#include "worms.h"
```

Include dependency graph for worms_screen.c:



Data Structures

- struct [game_window](#)

Structure to hold details of game windows.

Functions

- static void [worms_draw_title_window](#) (void)
- static void [worms_draw_info_window](#) (void)
- void [worms_game_area_init](#) (void)
- void [worms_game_area_destroy](#) (void)
- void [worms_draw_arena_border](#) (void)
- void [worms_draw_sidebar](#) (void)
- void [worms_refresh_game_area](#) (void)
- int [worms_game_arena_cols](#) (void)

Returns the number of columns in the game arena.

- int [worms_game_arena_rows](#) (void)

Returns the number of rows in the game arena.

- void [worms_write_arena_character](#) (const enum [worms_cell_character](#) ch, const int x, const int y)

Writes a character to the game arena.

- enum [worms_cell_character](#) [worms_get_arena_character](#) (const int x, const int y)

Returns the character at the specified arena coordinates.

- bool [worms_coords_in_game_arena](#) (const int x, const int y)

Tests if the specified coordinates are within the arena.

Variables

- static const int [sidebar_cols](#) = 20
- static const int [title_rows](#) = 7
- static struct [game_window](#) [arena_window](#)
- static struct [game_window](#) [title_window](#)
- static struct [game_window](#) [info_window](#)

5.17.1 Detailed Description

Implementation of worms game screen functions. The game screen consists of two parts: the "arena", where the worm moves, and the "sidebar", containing the game title and game statistics. "Game area" can also refer to the entire game screen.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.17.2 Function Documentation

5.17.2.1 `bool worms_coords_in_game_arena (const int x, const int y)`

Tests if the specified coordinates are within the arena. The coordinates are *not* considered to be within the arena if they are within the arena's border, so this test is suitable for determining whether the worm has hit the border at the edge of the arena.

Parameters

<code>x</code>	The specified x coordinate.
<code>y</code>	The specified y coordinate.

Returns

`true` if the specified coordinates are within the arena, excluding the arena border, and `false` otherwise.

5.17.2.2 `void worms_draw_arena_border (void)`

Draws a border around the game arena.

5.17.2.3 `static void worms_draw_info_window (void) [static]`

Draws the information window in the sidebar.

5.17.2.4 `void worms_draw_sidebar (void)`

Draws the sidebar.

5.17.2.5 `static void worms_draw_title_window (void) [static]`

Draws the title window in the sidebar.

5.17.2.6 `void worms_game_area_destroy (void)`

Destroys the game area, including the sidebar.

5.17.2.7 `void worms_game_area_init (void)`

Initializes the game area, including the sidebar.

5.17.2.8 `int worms_game_arena_cols (void)`

Returns the number of columns in the game arena. The number of columns *includes* the arena border, which is one character wide on all sides.

Returns

The number of columns in the game arena.

5.17.2.9 int worms_game_arena_rows (void)

Returns the number of rows in the game arena. The number of rows *includes* the arena border, which is one character wide on all sides.

Returns

The number of rows in the game arena.

5.17.2.10 enum worms_cell_character worms_get_arena_character (const int x, const int y)

Returns the character at the specified arena coordinates.

Parameters

<i>x</i>	The specified x coordinate.
<i>y</i>	The specified y coordinate.

Returns

The character at the specified coordinates in the arena.

5.17.2.11 void worms_refresh_game_area (void)

Refreshes the main arena and the sidebar.

5.17.2.12 void worms_write_arena_character (const enum worms_cell_character *ch*, const int x, const int y)

Writes a character to the game arena.

Parameters

<i>ch</i>	The character to write.
<i>x</i>	The x coordinate at which to write the character.
<i>y</i>	The y coordinate at which to write the character.

5.17.3 Variable Documentation

5.17.3.1 struct game_window arena_window [static]

File scope variable for main arena window

5.17.3.2 struct game_window info_window [static]

File scope variable for information window

5.17.3.3 const int sidebar_cols = 20 [static]

File scope variable for number of columns in sidebar

5.17.3.4 `const int title_rows = 7` [static]

File scope variable for number of rows in title window

5.17.3.5 `struct game_window title_window` [static]

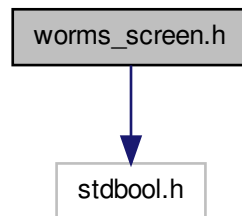
File scope variable for title window

5.18 `worms_screen.h` File Reference

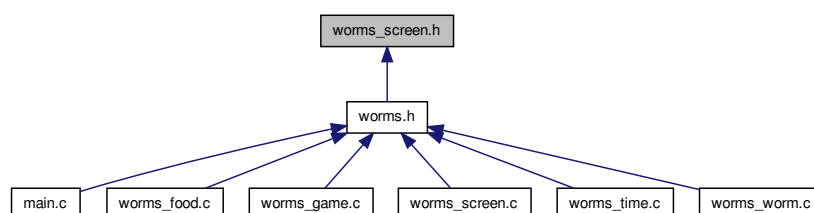
Interface to worms game screen functions.

```
#include <stdbool.h>
```

Include dependency graph for `worms_screen.h`:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `worms_cell_character` { `WORM_BODY_CHARACTER`, `WORM_EMPTY_CHARACTER`, `WORM_FOOD_CHARACTER` }

Functions

- void `worms_game_area_init` (void)
- void `worms_game_area_destroy` (void)

- void `worms_draw_arena_border` (void)
- void `worms_draw_sidebar` (void)
- void `worms_refresh_game_area` (void)
- int `worms_game_arena_cols` (void)
Returns the number of columns in the game arena.
- int `worms_game_arena_rows` (void)
Returns the number of rows in the game arena.
- void `worms_write_arena_character` (const enum `worms_cell_character` ch, const int x, const int y)
Writes a character to the game arena.
- enum `worms_cell_character` `worms_get_arena_character` (const int x, const int y)
Returns the character at the specified arena coordinates.
- bool `worms_coords_in_game_arena` (const int x, const int y)
Tests if the specified coordinates are within the arena.

5.18.1 Detailed Description

Interface to worms game screen functions. The game screen consists of two parts: the "arena", where the worm moves, and the "sidebar", containing the game title and game statistics. "Game area" can also refer to the entire game screen.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.18.2 Enumeration Type Documentation

5.18.2.1 enum worms_cell_character

Enumeration constants for game cell characters

Enumerator:

- `WORM_BODY_CHARACTER`** Character for worm body
- `WORM_EMPTY_CHARACTER`** Character for empty cell
- `WORM_FOOD_CHARACTER`** Character for piece of food

5.18.3 Function Documentation

5.18.3.1 bool worms_coords_in_game_arena (const int x, const int y)

Tests if the specified coordinates are within the arena. The coordinates are *not* considered to be within the arena if they are within the arena's border, so this test is suitable for determining whether the worm has hit the border at the edge of the arena.

Parameters

x	The specified x coordinate.
y	The specified y coordinate.

Returns

`true` if the specified coordinates are within the arena, excluding the arena border, and `false` otherwise.

5.18.3.2 void worms_draw_arena_border (void)

Draws a border around the game arena.

5.18.3.3 void worms_draw_sidebar (void)

Draws the sidebar.

5.18.3.4 void worms_game_area_destroy (void)

Destroys the game area, including the sidebar.

5.18.3.5 void worms_game_area_init (void)

Initializes the game area, including the sidebar.

5.18.3.6 int worms_game_arena_cols (void)

Returns the number of columns in the game arena. The number of columns *includes* the arena border, which is one character wide on all sides.

Returns

The number of columns in the game arena.

5.18.3.7 int worms_game_arena_rows (void)

Returns the number of rows in the game arena. The number of rows *includes* the arena border, which is one character wide on all sides.

Returns

The number of rows in the game arena.

5.18.3.8 enum worms_cell_character worms_get_arena_character (const int x, const int y)

Returns the character at the specified arena coordinates.

Parameters

<code>x</code>	The specified x coordinate.
<code>y</code>	The specified y coordinate.

Returns

The character at the specified coordinates in the arena.

5.18.3.9 void worms_refresh_game_area (void)

Refreshes the main arena and the sidebar.

5.18.3.10 void worms_write_arena_character (const enum worms_cell_character *ch*, const int *x*, const int *y*)

Writes a character to the game arena.

Parameters

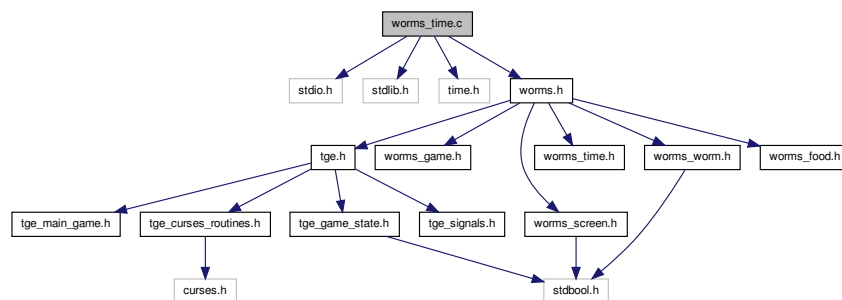
<i>ch</i>	The character to write.
<i>x</i>	The x coordinate at which to write the character.
<i>y</i>	The y coordinate at which to write the character.

5.19 worms_time.c File Reference

Implementation of worms game duration timer functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "worms.h"
```

Include dependency graph for worms_time.c:



Functions

- void [worms_time_init](#) (void)
- long [worms_game_time](#) (void)
Returns the number of seconds since the game started.
- char * [worms_game_time_string](#) (const bool long_format)
Returns a string representation of total game time.

Variables

- static time_t [start_time](#)

5.19.1 Detailed Description

Implementation of worms game duration timer functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.19.2 Function Documentation**5.19.2.1 `long worms_game_time (void)`**

Returns the number of seconds since the game started.

Returns

The number of seconds since the game started.

5.19.2.2 `char* worms_game_time_string (const bool long_format)`

Returns a string representation of total game time.

Parameters

<i>long_format</i>	If this parameter is <code>true</code> , the string representation is of the form "[H] hours, [M] minutes, and [S] seconds". If <code>false</code> , the string representation is of the form "HH:MM:SS".
--------------------	---

Returns

A string representation of the total game time. The returned pointer points to statically allocated storage, and is overwritten each time this function is called.

5.19.2.3 `void worms_time_init (void)`

Initializes the game duration timer.

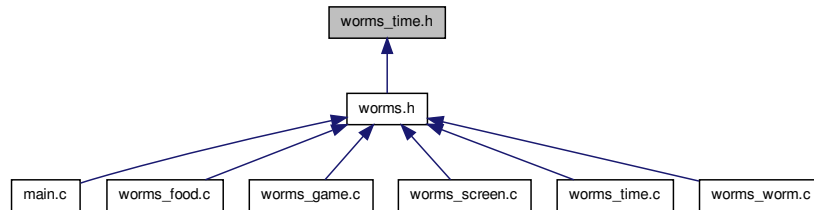
5.19.3 Variable Documentation**5.19.3.1 `time_t start_time [static]`**

File scope variable to hold the time at the start of the game

5.20 worms_time.h File Reference

Interface to worms game duration timer functions.

This graph shows which files directly or indirectly include this file:



Functions

- void [worms_time_init](#) (void)
- long [worms_game_time](#) (void)
Returns the number of seconds since the game started.
- char * [worms_game_time_string](#) (const bool long_format)
Returns a string representation of total game time.

5.20.1 Detailed Description

Interface to worms game duration timer functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.20.2 Function Documentation

5.20.2.1 long worms_game_time (void)

Returns the number of seconds since the game started.

Returns

The number of seconds since the game started.

5.20.2.2 char* worms_game_time_string (const bool long_format)

Returns a string representation of total game time.

Parameters

<i>long_format</i>	If this parameter is <code>true</code> , the string representation is of the form "[H] hours, [M] minutes, and [S] seconds". If <code>false</code> , the string representation is of the form "HH:MM:SS".
--------------------	---

Returns

A string representation of the total game time. The returned pointer points to statically allocated storage, and is overwritten each time this function is called.

5.20.2.3 void worms_time_init (void)

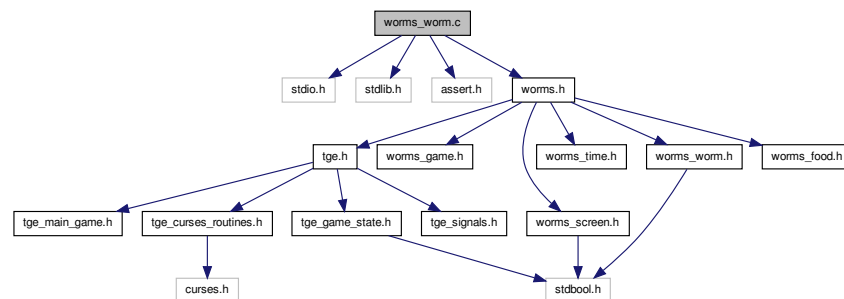
Initializes the game duration timer.

5.21 worms_worm.c File Reference

Implementation of worms game worm functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "worms.h"
```

Include dependency graph for worms_worm.c:



Data Structures

- struct [cell](#)
- struct [worm](#)

Functions

- static void [worm_clear](#) (void)
Erases the worm from the arena.
- static bool [worm_move](#) (void)
Moves the worm.
- static void [worm_draw](#) (void)
- static bool [worm_cell_here](#) (const int x, const int y)
Tests if the specified coordinates contain a worm body cell.
- static void [worm_add_cell_head](#) (const int x, const int y)
Adds a cell at the head of the worm.
- static void [worm_delete_cell_head](#) (void)
- static void [worm_delete_cell_tail](#) (void)
- void [worm_init](#) (void)
Initializes the worm.

- void `worm_destroy` (void)
Destroys the worm.
- void `worm_set_direction` (enum `worm_direction` direction)
Attempts to change the next direction of the worm.
- bool `worm_move_and_draw` (void)
Moves and draws the worm.

Variables

- static struct `worm worm` = {NULL, NULL, `WORM_DIR_RIGHT`, `WORM_DIR_RIGHT`}
- static const size_t `WORM_START_LENGTH` = 8

5.21.1 Detailed Description

Implementation of worms game worm functions. The worm is implemented as a double-ended, doubly-linked list.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.21.2 Function Documentation

5.21.2.1 static void worm_add_cell_head (const int x, const int y) [static]

Adds a cell at the head of the worm.

Parameters

<code>x</code>	The x coordinate of the new head.
<code>y</code>	The y coordinate of the new head.

5.21.2.2 static bool worm_cell_here (const int x, const int y) [static]

Tests if the specified coordinates contain a worm body cell. This function is used to test for collision with the worm's own body.

Parameters

<code>x</code>	The specified x coordinate.
<code>y</code>	The specified y coordinate.

Returns

`true` if the specified coordinates contain a worm body cell, `false` otherwise.

5.21.2.3 static void worm_clear (void) [static]

Erases the worm from the arena, i.e. overwrites it with empty cell characters.

5.21.2.4 `static void worm_delete_cell_head (void) [static]`

Deletes the cell at the head of the worm.

5.21.2.5 `static void worm_delete_cell_tail (void) [static]`

Deletes the cell at the tail of the worm.

5.21.2.6 `void worm_destroy (void)`

Destroys and deallocates memory for the worm.

5.21.2.7 `static void worm_draw (void) [static]`

Draws the worm.

5.21.2.8 `void worm_init (void)`

Initializes and allocates memory for the worm.

5.21.2.9 `static bool worm_move (void) [static]`

Moves the worm. The worm moves in the next direction specified in the worm structure. The function detects if the move would cause the worm to eat some food, and whether it would cause the worm to hit the arena wall, or its own body. In the former case, the tail cell is not deleted, and the worm grows in length. In the latter case, the game is ended.

Returns

`true` if the move caused the worm to eat some food, `false` in all other situations.

5.21.2.10 `bool worm_move_and_draw (void)`

Moves and draws the worm.

Returns

`true` if the movement that was performed caused the worm to eat a piece of food, `false` in all other situations.

5.21.2.11 `void worm_set_direction (enum worm_direction direction)`

Attempts to change the next direction of the worm. The worm will not actually change direction until the game next updates, and a second call to this function in the interim will nullify the effect of any previous calls, so it is safe to call this function an arbitrary number of times in between game updates. This function will ignore any attempt to change the direction in a way which is not allowed, particularly that the worm is not allowed to turn 180 degrees as this would cause it to hit its own body, so it is safe to call this function with any direction, and to trust it to ignore any direction changes which are not allowed.

Parameters

<i>direction</i>	The desired direction of the worm on the next update.
------------------	---

5.21.3 Variable Documentation

5.21.3.1 `struct worm worm = {NULL, NULL, WORM_DIR_RIGHT, WORM_DIR_RIGHT}` `[static]`

File scope variable to contain the worm

5.21.3.2 `const size_t WORM_START_LENGTH = 8` `[static]`

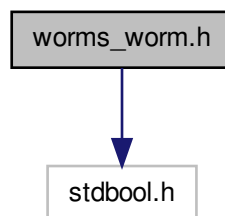
File scope constant containing initial length of worm

5.22 worms_worm.h File Reference

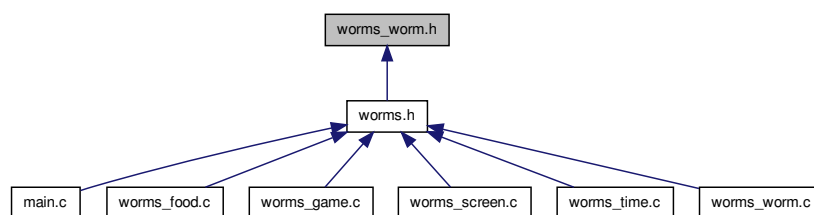
Interface to worms game worm functions.

```
#include <stdbool.h>
```

Include dependency graph for worms_worm.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `worm_direction` { `WORM_DIR_UP`, `WORM_DIR_RIGHT`, `WORM_DIR_DOWN`, `WORM_DIR_LEFT` }
Enumeration constants for worm movement direction.

Functions

- void `worm_init` (void)

- Initializes the worm.*
- void `worm_destroy` (void)
Destroys the worm.
- void `worm_set_direction` (enum `worm_direction` direction)
Attempts to change the next direction of the worm.
- bool `worm_move_and_draw` (void)
Moves and draws the worm.

5.22.1 Detailed Description

Interface to worms game worm functions.

Author

Paul Griffiths

Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

5.22.2 Enumeration Type Documentation

5.22.2.1 enum `worm_direction`

Enumeration constants for worm movement direction. The order in which these are defined is meaningful to the implementation, namely that pairs of opposing directions (i.e. up versus down, left versus right) have values with an absolute difference of 2.

Enumerator:

`WORM_DIR_UP` Up direction
`WORM_DIR_RIGHT` Right direction
`WORM_DIR_DOWN` Down direction
`WORM_DIR_LEFT` Left direction

5.22.3 Function Documentation

5.22.3.1 void `worm_destroy` (void)

Destroys and deallocates memory for the worm.

5.22.3.2 void `worm_init` (void)

Initializes and allocates memory for the worm.

5.22.3.3 bool `worm_move_and_draw` (void)

Moves and draws the worm.

Returns

`true` if the movement that was performed caused the worm to eat a piece of food, `false` in all other situations.

5.22.3.4 void worm_set_direction (enum worm_direction *direction*)

Attempts to change the next direction of the worm. The worm will not actually change direction until the game next updates, and a second call to this function in the interim will nullify the effect of any previous calls, so it is safe to call this function an arbitrary number of times in between game updates. This function will ignore any attempt to change the direction in a way which is not allowed, particularly that the worm is not allowed to turn 180 degrees as this would cause it to hit its own body, so it is safe to call this function with any direction, and to trust it to ignore any direction changes which are not allowed.

Parameters

<i>direction</i>	The desired direction of the worm on the next update.
------------------	---

Index

- `_POSIX_C_SOURCE`
 - `tge_curses_routines.c`, [17](#)
 - `tge_main_game.c`, [26](#)
 - `tge_signals.c`, [29](#)
- `arena_window`
 - `worms_screen.c`, [41](#)
- `back`
 - `cell`, [7](#)
- `cell`, [7](#)
 - `back`, [7](#)
 - `front`, [7](#)
 - `x`, [7](#)
 - `y`, [8](#)
- `cols`
 - `game_window`, [8](#)
- `draw_function`
 - `tge_parameters`, [9](#)
- `food_eaten`
 - `worms_food.c`, [33](#)
- `food_x`
 - `worms_food.c`, [34](#)
- `food_y`
 - `worms_food.c`, [34](#)
- `front`
 - `cell`, [7](#)
- `game_param`
 - `tge_main_game.c`, [26](#)
- `game_state`
 - `tge_game_state.c`, [21](#), [22](#)
- `game_window`, [8](#)
 - `cols`, [8](#)
 - `rows`, [8](#)
 - `tge_curses_routines.c`, [18](#)
 - `window`, [8](#)
- `gw`, [8](#)
 - `initialized`, [9](#)
 - `old_cursor`, [9](#)
 - `window`, [9](#)
 - `ws`, [9](#)
- `head`
 - `worm`, [10](#)
- `info_window`
 - `worms_screen.c`, [41](#)
- `initialized`
 - `gw`, [9](#)
- `input_function`
 - `tge_parameters`, [9](#)
- `last_direction`
 - `worm`, [10](#)
- `main`
 - `main.c`, [14](#)
- `main.c`, [13](#)
 - `main`, [14](#)
 - `print_quit_message`, [14](#)
- `next_direction`
 - `worm`, [11](#)
- `old_cursor`
 - `gw`, [9](#)
- `print_quit_message`
 - `main.c`, [14](#)
- `refresh_flag`
 - `tge_game_state.c`, [22](#)
- `rows`
 - `game_window`, [8](#)
- `setup_function`
 - `tge_parameters`, [9](#)
- `sidebar_cols`
 - `worms_screen.c`, [41](#)
- `start_time`
 - `worms_time.c`, [46](#)
- `TGE_GAME_STATE_ENDED`
 - `tge_game_state.c`, [21](#)
- `TGE_GAME_STATE_NOTSTARTED`
 - `tge_game_state.c`, [21](#)
- `TGE_GAME_STATE_RUNNING`
 - `tge_game_state.c`, [21](#)
- `tail`
 - `worm`, [11](#)
- `teardown_function`
 - `tge_parameters`, [9](#)
- `tge.h`, [14](#)
- `tge_game_state.c`
 - `TGE_GAME_STATE_ENDED`, [21](#)
 - `TGE_GAME_STATE_NOTSTARTED`, [21](#)
 - `TGE_GAME_STATE_RUNNING`, [21](#)
- `tge_begin_game`

- tge_main_game.c, 26
 - tge_main_game.h, 27
- tge_curses_routines.c, 15
 - game_window, 18
 - tge_free_screen, 17
 - tge_get_character, 17
 - tge_initialize_screen, 17
 - tge_main_window, 17
 - tge_term_cols, 17
 - tge_term_rows, 17
- tge_curses_routines.h, 18
 - tge_free_screen, 19
 - tge_get_character, 19
 - tge_initialize_screen, 19
 - tge_main_window, 19
 - tge_term_cols, 19
 - tge_term_rows, 19
- tge_end_game
 - tge_game_state.c, 21
 - tge_game_state.h, 24
 - tge_main_game.h, 27
- tge_end_status
 - tge_game_state.c, 21
 - tge_game_state.h, 24
- tge_end_status_var
 - tge_game_state.c, 22
- tge_free_screen
 - tge_curses_routines.c, 17
 - tge_curses_routines.h, 19
- tge_game_ended
 - tge_game_state.c, 21
 - tge_game_state.h, 24
- tge_game_loop
 - tge_main_game.c, 26
- tge_game_started
 - tge_game_state.c, 21
 - tge_game_state.h, 24
- tge_game_state.c, 20
 - game_state, 21, 22
 - refresh_flag, 22
 - tge_end_game, 21
 - tge_end_status, 21
 - tge_end_status_var, 22
 - tge_game_ended, 21
 - tge_game_started, 21
 - tge_needs_refresh, 22
 - tge_set_needs_refresh, 22
 - tge_start_game, 22
- tge_game_state.h, 22
 - tge_end_game, 24
 - tge_end_status, 24
 - tge_game_ended, 24
 - tge_game_started, 24
 - tge_needs_refresh, 24
 - tge_set_needs_refresh, 24
 - tge_start_game, 25
- tge_get_character
 - tge_curses_routines.c, 17
- tge_curses_routines.h, 19
- tge_get_fractional_microseconds
 - tge_signals.c, 29
- tge_get_whole_seconds
 - tge_signals.c, 29
- tge_handler
 - tge_signals.c, 29
- tge_initialize_screen
 - tge_curses_routines.c, 17
 - tge_curses_routines.h, 19
- tge_main_game.c, 25
 - game_param, 26
 - tge_begin_game, 26
 - tge_game_loop, 26
- tge_main_game.h, 26
 - tge_begin_game, 27
 - tge_end_game, 27
- tge_main_window
 - tge_curses_routines.c, 17
 - tge_curses_routines.h, 19
- tge_needs_refresh
 - tge_game_state.c, 22
 - tge_game_state.h, 24
- tge_parameters, 9
 - draw_function, 9
 - input_function, 9
 - setup_function, 9
 - teardown_function, 9
 - timer_interval, 10
- tge_set_needs_refresh
 - tge_game_state.c, 22
 - tge_game_state.h, 24
- tge_set_signal_handlers
 - tge_signals.c, 30
 - tge_signals.h, 31
- tge_signals.c, 28
 - _POSIX_C_SOURCE, 29
 - tge_get_fractional_microseconds, 29
 - tge_get_whole_seconds, 29
 - tge_handler, 29
 - tge_set_signal_handlers, 30
 - tge_timer_start, 30
 - tge_timer_stop, 30
- tge_signals.h, 30
 - tge_set_signal_handlers, 31
 - tge_timer_start, 31
 - tge_timer_stop, 31
- tge_start_game
 - tge_game_state.c, 22
 - tge_game_state.h, 25
- tge_term_cols
 - tge_curses_routines.c, 17
 - tge_curses_routines.h, 19
- tge_term_rows
 - tge_curses_routines.c, 17
 - tge_curses_routines.h, 19
- tge_timer_start
 - tge_signals.c, 30

- tge_signals.h, [31](#)
- tge_timer_stop
 - tge_signals.c, [30](#)
 - tge_signals.h, [31](#)
- timer_interval
 - tge_parameters, [10](#)
- title_rows
 - worms_screen.c, [41](#)
- title_window
 - worms_screen.c, [42](#)
- WORM_BODY_CHARACTER
 - worms_screen.h, [43](#)
- WORM_DIR_DOWN
 - worms_worm.h, [52](#)
- WORM_DIR_LEFT
 - worms_worm.h, [52](#)
- WORM_DIR_RIGHT
 - worms_worm.h, [52](#)
- WORM_DIR_UP
 - worms_worm.h, [52](#)
- WORM_EMPTY_CHARACTER
 - worms_screen.h, [43](#)
- WORM_FOOD_CHARACTER
 - worms_screen.h, [43](#)
- WORMS_EXIT_HITSELF
 - worms_game.h, [38](#)
- WORMS_EXIT_HITWALL
 - worms_game.h, [37](#)
- WORMS_EXIT_NORMAL
 - worms_game.h, [37](#)
- WORM_START_LENGTH
 - worms_worm.c, [51](#)
- window
 - game_window, [8](#)
 - gw, [9](#)
- worm, [10](#)
 - head, [10](#)
 - last_direction, [10](#)
 - next_direction, [11](#)
 - tail, [11](#)
 - worms_worm.c, [51](#)
- worm_add_cell_head
 - worms_worm.c, [49](#)
- worm_cell_here
 - worms_worm.c, [49](#)
- worm_clear
 - worms_worm.c, [49](#)
- worm_delete_cell_head
 - worms_worm.c, [49](#)
- worm_delete_cell_tail
 - worms_worm.c, [50](#)
- worm_destroy
 - worms_worm.c, [50](#)
 - worms_worm.h, [52](#)
- worm_direction
 - worms_worm.h, [52](#)
- worm_draw
 - worms_worm.c, [50](#)
- worm_init
 - worms_worm.c, [50](#)
 - worms_worm.h, [52](#)
- worm_move
 - worms_worm.c, [50](#)
- worm_move_and_draw
 - worms_worm.c, [50](#)
 - worms_worm.h, [52](#)
- worm_set_direction
 - worms_worm.c, [50](#)
 - worms_worm.h, [52](#)
- worms.dox, [31](#)
- worms.h, [31](#)
- worms_game.h
 - WORMS_EXIT_HITSELF, [38](#)
 - WORMS_EXIT_HITWALL, [37](#)
 - WORMS_EXIT_NORMAL, [37](#)
- worms_screen.h
 - WORM_BODY_CHARACTER, [43](#)
 - WORM_EMPTY_CHARACTER, [43](#)
 - WORM_FOOD_CHARACTER, [43](#)
- worms_worm.h
 - WORM_DIR_DOWN, [52](#)
 - WORM_DIR_LEFT, [52](#)
 - WORM_DIR_RIGHT, [52](#)
 - WORM_DIR_UP, [52](#)
- worms_cell_character
 - worms_screen.h, [43](#)
- worms_coords_in_game_arena
 - worms_screen.c, [40](#)
 - worms_screen.h, [43](#)
- worms_draw_arena_border
 - worms_screen.c, [40](#)
 - worms_screen.h, [44](#)
- worms_draw_food
 - worms_food.c, [33](#)
 - worms_food.h, [35](#)
- worms_draw_info_window
 - worms_screen.c, [40](#)
- worms_draw_screen
 - worms_game.c, [36](#)
 - worms_game.h, [38](#)
- worms_draw_sidebar
 - worms_screen.c, [40](#)
 - worms_screen.h, [44](#)
- worms_draw_title_window
 - worms_screen.c, [40](#)
- worms_exit_status
 - worms_game.h, [37](#)
- worms_food.c, [32](#)
 - food_eaten, [33](#)
 - food_x, [34](#)
 - food_y, [34](#)
 - worms_draw_food, [33](#)
 - worms_food_here, [33](#)
 - worms_get_food_eaten, [33](#)
 - worms_place_new_food, [33](#)
- worms_food.h, [34](#)

- worms_draw_food, 35
- worms_food_here, 35
- worms_get_food_eaten, 35
- worms_place_new_food, 35
- worms_food_here
 - worms_food.c, 33
 - worms_food.h, 35
- worms_game.c, 35
 - worms_draw_screen, 36
 - worms_game_setup, 36
 - worms_game_teardown, 36
 - worms_process_input, 36
- worms_game.h, 37
 - worms_draw_screen, 38
 - worms_exit_status, 37
 - worms_game_setup, 38
 - worms_game_teardown, 38
 - worms_process_input, 38
- worms_game_area_destroy
 - worms_screen.c, 40
 - worms_screen.h, 44
- worms_game_area_init
 - worms_screen.c, 40
 - worms_screen.h, 44
- worms_game_arena_cols
 - worms_screen.c, 40
 - worms_screen.h, 44
- worms_game_arena_rows
 - worms_screen.c, 40
 - worms_screen.h, 44
- worms_game_setup
 - worms_game.c, 36
 - worms_game.h, 38
- worms_game_teardown
 - worms_game.c, 36
 - worms_game.h, 38
- worms_game_time
 - worms_time.c, 46
 - worms_time.h, 47
- worms_game_time_string
 - worms_time.c, 46
 - worms_time.h, 47
- worms_get_arena_character
 - worms_screen.c, 41
 - worms_screen.h, 44
- worms_get_food_eaten
 - worms_food.c, 33
 - worms_food.h, 35
- worms_place_new_food
 - worms_food.c, 33
 - worms_food.h, 35
- worms_process_input
 - worms_game.c, 36
 - worms_game.h, 38
- worms_refresh_game_area
 - worms_screen.c, 41
 - worms_screen.h, 44
- worms_screen.c, 38
- arena_window, 41
- info_window, 41
- sidebar_cols, 41
- title_rows, 41
- title_window, 42
- worms_coords_in_game_arena, 40
- worms_draw_arena_border, 40
- worms_draw_info_window, 40
- worms_draw_sidebar, 40
- worms_draw_title_window, 40
- worms_game_area_destroy, 40
- worms_game_area_init, 40
- worms_game_arena_cols, 40
- worms_game_arena_rows, 40
- worms_get_arena_character, 41
- worms_refresh_game_area, 41
- worms_write_arena_character, 41
- worms_screen.h, 42
 - worms_cell_character, 43
 - worms_coords_in_game_arena, 43
 - worms_draw_arena_border, 44
 - worms_draw_sidebar, 44
 - worms_game_area_destroy, 44
 - worms_game_area_init, 44
 - worms_game_arena_cols, 44
 - worms_game_arena_rows, 44
 - worms_get_arena_character, 44
 - worms_refresh_game_area, 44
 - worms_write_arena_character, 45
- worms_time.c, 45
 - start_time, 46
 - worms_game_time, 46
 - worms_game_time_string, 46
 - worms_time_init, 46
- worms_time.h, 46
 - worms_game_time, 47
 - worms_game_time_string, 47
 - worms_time_init, 48
- worms_time_init
 - worms_time.c, 46
 - worms_time.h, 48
- worms_worm.c, 48
 - WORM_START_LENGTH, 51
 - worm, 51
 - worm_add_cell_head, 49
 - worm_cell_here, 49
 - worm_clear, 49
 - worm_delete_cell_head, 49
 - worm_delete_cell_tail, 50
 - worm_destroy, 50
 - worm_draw, 50
 - worm_init, 50
 - worm_move, 50
 - worm_move_and_draw, 50
 - worm_set_direction, 50
- worms_worm.h, 51
 - worm_destroy, 52
 - worm_direction, 52

- worm_init, [52](#)
- worm_move_and_draw, [52](#)
- worm_set_direction, [52](#)
- worms_write_arena_character
 - worms_screen.c, [41](#)
 - worms_screen.h, [45](#)
- ws
 - gw, [9](#)
- x
 - cell, [7](#)
- y
 - cell, [8](#)