

tAltris  
v1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	ai_coefs Struct Reference . . . . .	5
3.1.1	Field Documentation . . . . .	5
3.1.1.1	agg_height . . . . .	5
3.1.1.2	bumpiness . . . . .	5
3.1.1.3	clears . . . . .	5
3.1.1.4	holes . . . . .	6
3.2	board Struct Reference . . . . .	6
3.2.1	Field Documentation . . . . .	6
3.2.1.1	data . . . . .	6
3.3	game_state Struct Reference . . . . .	6
3.3.1	Field Documentation . . . . .	7
3.3.1.1	board . . . . .	7
3.3.1.2	broken_lines . . . . .	7
3.3.1.3	level . . . . .	7
3.3.1.4	piece_current . . . . .	7
3.3.1.5	piece_next . . . . .	7
3.3.1.6	score . . . . .	8

3.3.1.7	state	8
3.3.1.8	time	8
3.4	input Struct Reference	8
3.4.1	Field Documentation	8
3.4.1.1	down	8
3.4.1.2	moveX	9
3.4.1.3	moveY	9
3.4.1.4	quit	9
3.4.1.5	rotate	9
3.5	list Struct Reference	9
3.5.1	Detailed Description	10
3.5.2	Field Documentation	10
3.5.2.1	first	10
3.5.2.2	length	10
3.6	list_node Struct Reference	10
3.6.1	Detailed Description	11
3.6.2	Field Documentation	11
3.6.2.1	next	11
3.7	matrix Struct Reference	11
3.7.1	Detailed Description	11
3.7.2	Field Documentation	11
3.7.2.1	cols	12
3.7.2.2	data	12
3.7.2.3	rows	12
3.8	piece Struct Reference	12
3.8.1	Field Documentation	12
3.8.1.1	angle	12
3.8.1.2	id	13
3.8.1.3	shapes	13
3.8.1.4	x	13
3.8.1.5	y	13

<b>4 File Documentation</b>	<b>15</b>
4.1 src/ai/genetic/engine.c File Reference . . . . .	15
4.2 src/ai/genetic/engine.h File Reference . . . . .	16
4.3 src/ai/genetic/tools.c File Reference . . . . .	17
4.3.1 Function Documentation . . . . .	17
4.3.1.1 aggregate_height() . . . . .	17
4.3.1.2 board_height() . . . . .	18
4.3.1.3 board_heights() . . . . .	18
4.3.1.4 bumpiness() . . . . .	18
4.3.1.5 clears() . . . . .	18
4.3.1.6 hole() . . . . .	18
4.3.1.7 holes() . . . . .	18
4.3.1.8 make_line() . . . . .	19
4.4 src/ai/genetic/tools.h File Reference . . . . .	19
4.4.1 Macro Definition Documentation . . . . .	20
4.4.1.1 ABS . . . . .	20
4.4.2 Function Documentation . . . . .	20
4.4.2.1 aggregate_height() . . . . .	21
4.4.2.2 board_height() . . . . .	21
4.4.2.3 board_heights() . . . . .	21
4.4.2.4 bumpiness() . . . . .	21
4.4.2.5 clears() . . . . .	21
4.4.2.6 coalescent_clears() . . . . .	21
4.4.2.7 hole() . . . . .	22
4.4.2.8 holes() . . . . .	22
4.4.2.9 make_line() . . . . .	22
4.5 src/core/board.c File Reference . . . . .	22
4.5.1 Detailed Description . . . . .	23
4.5.2 Function Documentation . . . . .	23
4.5.2.1 board_at() . . . . .	23

4.5.2.2	board_break_line()	23
4.5.2.3	board_break_lines()	24
4.5.2.4	board_check_position()	24
4.5.2.5	board_copy()	24
4.5.2.6	board_create()	24
4.5.2.7	board_free()	24
4.5.2.8	board_get_completed_lines()	24
4.5.2.9	board_init()	24
4.5.2.10	board_is_line_complete()	25
4.5.2.11	board_merge_piece()	25
4.5.2.12	board_move_line()	25
4.5.2.13	board_print()	25
4.5.2.14	board_remove_line()	25
4.5.2.15	board_set()	25
4.6	src/core/board.h File Reference	26
4.6.1	Detailed Description	27
4.6.2	Macro Definition Documentation	27
4.6.2.1	BOARD_CELL_EMPTY	27
4.6.2.2	BOARD_HEIGHT	28
4.6.2.3	BOARD_WIDTH	28
4.6.3	Function Documentation	28
4.6.3.1	board_at()	28
4.6.3.2	board_break_line()	28
4.6.3.3	board_break_lines()	28
4.6.3.4	board_check_position()	28
4.6.3.5	board_copy()	29
4.6.3.6	board_create()	29
4.6.3.7	board_free()	29
4.6.3.8	board_get_completed_lines()	29
4.6.3.9	board_init()	29

4.6.3.10	board_is_line_complete()	29
4.6.3.11	board_merge_piece()	29
4.6.3.12	board_move_line()	30
4.6.3.13	board_print()	30
4.6.3.14	board_remove_line()	30
4.6.3.15	board_set()	30
4.7	src/core/event.c File Reference	30
4.7.1	Detailed Description	31
4.7.2	Function Documentation	31
4.7.2.1	event_handle()	32
4.8	src/core/event.h File Reference	32
4.8.1	Detailed Description	33
4.8.2	Function Documentation	33
4.8.2.1	event_handle()	33
4.9	src/core/game.c File Reference	34
4.9.1	Detailed Description	34
4.9.2	Function Documentation	34
4.9.2.1	game_tick()	35
4.10	src/core/game.h File Reference	35
4.10.1	Detailed Description	36
4.10.2	Function Documentation	36
4.10.2.1	game_tick()	36
4.11	src/core/game_state.c File Reference	37
4.11.1	Detailed Description	37
4.11.2	Function Documentation	37
4.11.2.1	gs_create()	38
4.11.2.2	gs_free()	38
4.11.2.3	gs_init()	38
4.11.2.4	gs_next_piece()	38
4.12	src/core/game_state.h File Reference	38

4.12.1 Detailed Description . . . . .	39
4.12.2 Macro Definition Documentation . . . . .	39
4.12.2.1 GS_SPAWN_X . . . . .	40
4.12.2.2 GS_SPAWN_Y . . . . .	40
4.12.2.3 GS_STATE_GAMEOVER . . . . .	40
4.12.2.4 GS_STATE_PAUSED . . . . .	40
4.12.2.5 GS_STATE_PLAYING . . . . .	40
4.12.2.6 GS_STATE_QUIT . . . . .	40
4.12.3 Function Documentation . . . . .	40
4.12.3.1 gs_create() . . . . .	40
4.12.3.2 gs_free() . . . . .	41
4.12.3.3 gs_init() . . . . .	41
4.12.3.4 gs_next_piece() . . . . .	41
4.13 src/core/input.c File Reference . . . . .	41
4.13.1 Detailed Description . . . . .	41
4.14 src/core/input.h File Reference . . . . .	42
4.14.1 Detailed Description . . . . .	42
4.15 src/core/motion.c File Reference . . . . .	42
4.15.1 Detailed Description . . . . .	43
4.15.2 Function Documentation . . . . .	43
4.15.2.1 motion_can_move() . . . . .	44
4.15.2.2 motion_can_rotate() . . . . .	44
4.15.2.3 motion_try_move() . . . . .	44
4.15.2.4 motion_try_move_down() . . . . .	44
4.15.2.5 motion_try_rotate() . . . . .	44
4.16 src/core/motion.h File Reference . . . . .	45
4.16.1 Detailed Description . . . . .	46
4.16.2 Function Documentation . . . . .	46
4.16.2.1 motion_can_move() . . . . .	46
4.16.2.2 motion_can_rotate() . . . . .	46



4.16.2.3	<code>motion_try_move()</code> . . . . .	46
4.16.2.4	<code>motion_try_move_down()</code> . . . . .	47
4.16.2.5	<code>motion_try_rotate()</code> . . . . .	47
4.17	<code>src/core/piece.c</code> File Reference . . . . .	47
4.17.1	Detailed Description . . . . .	48
4.17.2	Function Documentation . . . . .	48
4.17.2.1	<code>piece_move()</code> . . . . .	48
4.17.2.2	<code>piece_random()</code> . . . . .	48
4.17.2.3	<code>piece_rotate()</code> . . . . .	48
4.17.3	Variable Documentation . . . . .	48
4.17.3.1	<code>PIECE_SHAPES</code> . . . . .	49
4.18	<code>src/core/piece.h</code> File Reference . . . . .	49
4.18.1	Detailed Description . . . . .	50
4.18.2	Macro Definition Documentation . . . . .	50
4.18.2.1	<code>PIECE_ANGLE_DOWN</code> . . . . .	51
4.18.2.2	<code>PIECE_ANGLE_LEFT</code> . . . . .	51
4.18.2.3	<code>PIECE_ANGLE_RIGHT</code> . . . . .	51
4.18.2.4	<code>PIECE_ANGLE_UP</code> . . . . .	51
4.18.2.5	<code>PIECE_ANGLES</code> . . . . .	51
4.18.2.6	<code>PIECE_COUNT</code> . . . . .	51
4.18.2.7	<code>PIECE_HEIGHT</code> . . . . .	51
4.18.2.8	<code>PIECE_I</code> . . . . .	51
4.18.2.9	<code>PIECE_J</code> . . . . .	52
4.18.2.10	<code>PIECE_L</code> . . . . .	52
4.18.2.11	<code>PIECE_O</code> . . . . .	52
4.18.2.12	<code>PIECE_ROTATE_LEFT</code> . . . . .	52
4.18.2.13	<code>PIECE_ROTATE_RIGHT</code> . . . . .	52
4.18.2.14	<code>PIECE_S</code> . . . . .	52
4.18.2.15	<code>PIECE_T</code> . . . . .	52
4.18.2.16	<code>PIECE_WIDTH</code> . . . . .	52

4.18.2.17	PIECE_Z . . . . .	53
4.18.3	Function Documentation . . . . .	53
4.18.3.1	piece_move() . . . . .	53
4.18.3.2	piece_random() . . . . .	53
4.18.3.3	piece_rotate() . . . . .	53
4.18.4	Variable Documentation . . . . .	53
4.18.4.1	PIECE_SHAPES . . . . .	53
4.19	src/tAltris.c File Reference . . . . .	54
4.19.1	Detailed Description . . . . .	54
4.19.2	Function Documentation . . . . .	54
4.19.2.1	main() . . . . .	54
4.20	src/tAltris.h File Reference . . . . .	55
4.20.1	Detailed Description . . . . .	55
4.21	src/ui/gui.c File Reference . . . . .	56
4.21.1	Detailed Description . . . . .	56
4.21.2	Function Documentation . . . . .	57
4.21.2.1	gui_free() . . . . .	57
4.21.2.2	gui_init() . . . . .	57
4.21.2.3	gui_load_image() . . . . .	57
4.22	src/ui/gui.h File Reference . . . . .	57
4.22.1	Detailed Description . . . . .	58
4.22.2	Macro Definition Documentation . . . . .	58
4.22.2.1	GUI_HEIGHT . . . . .	59
4.22.2.2	GUI_TITLE . . . . .	59
4.22.2.3	GUI_WIDTH . . . . .	59
4.22.3	Function Documentation . . . . .	59
4.22.3.1	gui_free() . . . . .	59
4.22.3.2	gui_init() . . . . .	59
4.22.3.3	gui_load_image() . . . . .	59
4.23	src/ui/render.c File Reference . . . . .	60

4.23.1 Detailed Description . . . . .	60
4.23.2 Function Documentation . . . . .	61
4.23.2.1 render_board() . . . . .	61
4.23.2.2 render_handle() . . . . .	61
4.23.2.3 render_next_piece() . . . . .	61
4.23.2.4 render_piece() . . . . .	61
4.24 src/ui/render.h File Reference . . . . .	61
4.24.1 Detailed Description . . . . .	63
4.24.2 Macro Definition Documentation . . . . .	63
4.24.2.1 RENDER_CELL_SIZE . . . . .	63
4.24.2.2 RENDER_FPS . . . . .	63
4.24.3 Function Documentation . . . . .	63
4.24.3.1 render_board() . . . . .	63
4.24.3.2 render_handle() . . . . .	64
4.24.3.3 render_next_piece() . . . . .	64
4.24.3.4 render_piece() . . . . .	64
4.25 src/utls/list.c File Reference . . . . .	64
4.25.1 Detailed Description . . . . .	65
4.25.2 Function Documentation . . . . .	65
4.25.2.1 list_add() . . . . .	65
4.25.2.2 list_advance() . . . . .	66
4.25.2.3 list_append() . . . . .	67
4.25.2.4 list_at() . . . . .	68
4.25.2.5 list_concat() . . . . .	69
4.25.2.6 list_del() . . . . .	69
4.25.2.7 list_del_after() . . . . .	70
4.25.2.8 list_del_at() . . . . .	70
4.25.2.9 list_first() . . . . .	71
4.25.2.10 list_init() . . . . .	71
4.25.2.11 list_insert_after() . . . . .	72

4.25.2.12	list_insert_at()	73
4.25.2.13	list_is_empty()	73
4.25.2.14	list_last()	74
4.25.2.15	list_length()	74
4.25.2.16	list_next()	75
4.25.2.17	list_print()	75
4.25.2.18	list_reverse()	75
4.25.2.19	list_sort()	77
4.25.2.20	list_split_at()	78
4.25.2.21	list_swap()	78
4.26	src/utls/list.h File Reference	79
4.26.1	Detailed Description	80
4.26.2	Macro Definition Documentation	81
4.26.2.1	list_elt	81
4.26.2.2	list_foreach	81
4.26.2.3	list_foreach_elt	82
4.26.2.4	list_foreach_elt_safe	82
4.26.2.5	list_foreach_safe	83
4.26.3	Function Documentation	84
4.26.3.1	list_add()	84
4.26.3.2	list_advance()	84
4.26.3.3	list_append()	85
4.26.3.4	list_at()	86
4.26.3.5	list_concat()	87
4.26.3.6	list_del()	88
4.26.3.7	list_del_after()	88
4.26.3.8	list_del_at()	89
4.26.3.9	list_first()	89
4.26.3.10	list_init()	90
4.26.3.11	list_insert_after()	90

4.26.3.12 list_insert_at()	91
4.26.3.13 list_is_empty()	91
4.26.3.14 list_last()	92
4.26.3.15 list_length()	92
4.26.3.16 list_next()	93
4.26.3.17 list_print()	93
4.26.3.18 list_reverse()	94
4.26.3.19 list_sort()	94
4.26.3.20 list_split_at()	95
4.26.3.21 list_swap()	96
4.27 src/utls/matrix.c File Reference	96
4.27.1 Detailed Description	97
4.27.2 Function Documentation	97
4.27.2.1 matrix_at()	97
4.27.2.2 matrix_cols()	98
4.27.2.3 matrix_copy()	98
4.27.2.4 matrix_create()	99
4.27.2.5 matrix_create_from_array()	100
4.27.2.6 matrix_diagonal()	101
4.27.2.7 matrix_dot_product()	102
4.27.2.8 matrix_free()	102
4.27.2.9 matrix_hadamard_product()	103
4.27.2.10 matrix_identity()	103
4.27.2.11 matrix_is_diagonal()	104
4.27.2.12 matrix_is_square()	104
4.27.2.13 matrix_is_upper_triangularized()	105
4.27.2.14 matrix_print()	105
4.27.2.15 matrix_product()	106
4.27.2.16 matrix_rows()	107
4.27.2.17 matrix_scale()	107

4.27.2.18	<code>matrix_set()</code>	108
4.27.2.19	<code>matrix_sum()</code>	108
4.27.2.20	<code>matrix_transpose()</code>	109
4.28	<code>src/utils/matrix.h</code> File Reference	110
4.28.1	Detailed Description	111
4.28.2	Function Documentation	111
4.28.2.1	<code>matrix_at()</code>	111
4.28.2.2	<code>matrix_cols()</code>	112
4.28.2.3	<code>matrix_copy()</code>	112
4.28.2.4	<code>matrix_create()</code>	113
4.28.2.5	<code>matrix_create_from_array()</code>	113
4.28.2.6	<code>matrix_diagonal()</code>	114
4.28.2.7	<code>matrix_dot_product()</code>	115
4.28.2.8	<code>matrix_free()</code>	115
4.28.2.9	<code>matrix_hadamard_product()</code>	116
4.28.2.10	<code>matrix_identity()</code>	116
4.28.2.11	<code>matrix_is_diagonal()</code>	117
4.28.2.12	<code>matrix_is_square()</code>	117
4.28.2.13	<code>matrix_is_upper_triangularized()</code>	118
4.28.2.14	<code>matrix_print()</code>	118
4.28.2.15	<code>matrix_product()</code>	119
4.28.2.16	<code>matrix_rows()</code>	120
4.28.2.17	<code>matrix_scale()</code>	120
4.28.2.18	<code>matrix_set()</code>	121
4.28.2.19	<code>matrix_sum()</code>	121
4.28.2.20	<code>matrix_transpose()</code>	122
4.29	<code>src/utils/random.c</code> File Reference	123
4.29.1	Detailed Description	123
4.29.2	Function Documentation	123
4.29.2.1	<code>random_init()</code>	124

4.29.2.2	random_int()	124
4.29.2.3	random_size_t()	124
4.30	src/utls/random.h File Reference	124
4.30.1	Detailed Description	125
4.30.2	Function Documentation	125
4.30.2.1	random_init()	125
4.30.2.2	random_int()	125
4.30.2.3	random_size_t()	126
4.31	src/utls/timing.c File Reference	126
4.31.1	Detailed Description	126
4.31.2	Function Documentation	127
4.31.2.1	time_get_current()	127
4.32	src/utls/timing.h File Reference	127
4.32.1	Detailed Description	128
4.32.2	Function Documentation	128
4.32.2.1	time_get_current()	128
<b>Index</b>		<b>129</b>





## Chapter 1

# Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<b>ai_coefs</b>	5
<b>board</b>	6
<b>game_state</b>	6
<b>input</b>	8
<b>list</b>	9
<b>list_node</b>	10
<b>matrix</b>	11
<b>piece</b>	12



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

src/ <b>tAltris.c</b>	
Main file . . . . .	54
src/ <b>tAltris.h</b>	
Main file . . . . .	55
src/ai/genetic/ <b>engine.c</b>	15
src/ai/genetic/ <b>engine.h</b>	16
src/ai/genetic/ <b>tools.c</b>	17
src/ai/genetic/ <b>tools.h</b>	19
src/core/ <b>board.c</b>	
No description . . . . .	22
src/core/ <b>board.h</b>	
No description . . . . .	26
src/core/ <b>event.c</b>	
No description . . . . .	30
src/core/ <b>event.h</b>	
No description . . . . .	32
src/core/ <b>game.c</b>	
No description . . . . .	34
src/core/ <b>game.h</b>	
No description . . . . .	35
src/core/ <b>game_state.c</b>	
No description . . . . .	37
src/core/ <b>game_state.h</b>	
No description . . . . .	38
src/core/ <b>input.c</b>	
No description . . . . .	41
src/core/ <b>input.h</b>	
No description . . . . .	42
src/core/ <b>motion.c</b>	
No description . . . . .	42
src/core/ <b>motion.h</b>	
No description . . . . .	45
src/core/ <b>piece.c</b>	
No description . . . . .	47
src/core/ <b>piece.h</b>	
No description . . . . .	49

src/ui/ <b>gui.c</b>	
No description	56
src/ui/ <b>gui.h</b>	
No description	57
src/ui/ <b>render.c</b>	
No description	60
src/ui/ <b>render.h</b>	
No description	61
src/utls/ <b>list.c</b>	
Intrusive list implement	64
src/utls/ <b>list.h</b>	
Intrusive list implement	79
src/utls/ <b>matrix.c</b>	
Matrix implement	96
src/utls/ <b>matrix.h</b>	
Matrix implement	110
src/utls/ <b>random.c</b>	
No description	123
src/utls/ <b>random.h</b>	
No description	124
src/utls/ <b>timing.c</b>	
No description	126
src/utls/ <b>timing.h</b>	
No description	127

## Chapter 3

# Data Structure Documentation

### 3.1 ai\_coefs Struct Reference

```
#include <engine.h>
```

#### Data Fields

- double **agg\_height**
- double **holes**
- double **clears**
- double **bumpiness**

#### 3.1.1 Field Documentation

##### 3.1.1.1 agg\_height

```
double agg_height
```

##### 3.1.1.2 bumpiness

```
double bumpiness
```

##### 3.1.1.3 clears

```
double clears
```

#### 3.1.1.4 holes

```
double holes
```

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

- `src/ai/genetic/ engine.h`

## 3.2 board Struct Reference

```
#include <board.h>
```

### Data Fields

- `int data [ BOARD_HEIGHT][ BOARD_WIDTH]`

#### 3.2.1 Field Documentation

##### 3.2.1.1 data

```
int data[ BOARD_HEIGHT][ BOARD_WIDTH]
```

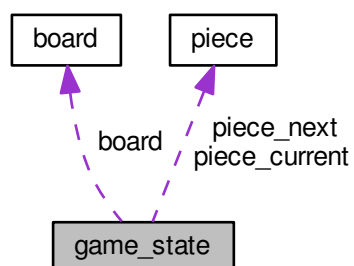
The documentation for this struct was generated from the following file:

- `src/core/ board.h`

## 3.3 game\_state Struct Reference

```
#include <game_state.h>
```

Collaboration diagram for game\_state:



## Data Fields

- int **level**
- int **score**
- int **broken\_lines**
- int **state**
- double **time**
- struct **piece** **piece\_current**
- struct **piece** **piece\_next**
- struct **board** \* **board**

### 3.3.1 Field Documentation

#### 3.3.1.1 board

```
struct board* board
```

#### 3.3.1.2 broken\_lines

```
int broken_lines
```

#### 3.3.1.3 level

```
int level
```

#### 3.3.1.4 piece\_current

```
struct piece piece_current
```

#### 3.3.1.5 piece\_next

```
struct piece piece_next
```

#### 3.3.1.6 score

```
int score
```

#### 3.3.1.7 state

```
int state
```

#### 3.3.1.8 time

```
double time
```

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

- `src/core/ game_state.h`

## 3.4 input Struct Reference

```
#include <input.h>
```

### Data Fields

- int **moveX**
- int **moveY**
- int **rotate**
- int **down**
- int **quit**

#### 3.4.1 Field Documentation

##### 3.4.1.1 down

```
int down
```



#### 3.4.1.2 moveX

```
int moveX
```

#### 3.4.1.3 moveY

```
int moveY
```

#### 3.4.1.4 quit

```
int quit
```

#### 3.4.1.5 rotate

```
int rotate
```

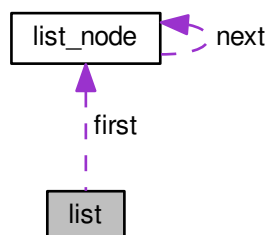
The documentation for this struct was generated from the following file:

- `src/core/ input.h`

## 3.5 list Struct Reference

```
#include <list.h>
```

Collaboration diagram for list:



## Data Fields

- `size_t` **length**
- `struct list_node *` **first**

### 3.5.1 Detailed Description

Head of a singly-linked list.

### 3.5.2 Field Documentation

#### 3.5.2.1 first

```
struct list_node* first
```

First node.

#### 3.5.2.2 length

```
size_t length
```

List length.

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

- `src/utils/` **list.h**

## 3.6 list\_node Struct Reference

```
#include <list.h>
```

Collaboration diagram for list\_node:



## Data Fields

- struct **list\_node** \* **next**

### 3.6.1 Detailed Description

A node of a singly-linked list.

### 3.6.2 Field Documentation

#### 3.6.2.1 next

```
struct list_node* next
```

Next node.

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

- src/utlis/ **list.h**

## 3.7 matrix Struct Reference

```
#include <matrix.h>
```

## Data Fields

- size\_t **rows**
- size\_t **cols**
- double \* **data**

### 3.7.1 Detailed Description

Matrix structure

### 3.7.2 Field Documentation

### 3.7.2.1 cols

```
size_t cols
```

Columns

### 3.7.2.2 data

```
double* data
```

Values

### 3.7.2.3 rows

```
size_t rows
```

Rows

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

- src/utils/ **matrix.h**

## 3.8 piece Struct Reference

```
#include <piece.h>
```

### Data Fields

- size\_t **id**
- size\_t **x**
- size\_t **y**
- size\_t **angle**
- int **shapes** [ **PIECE\_ANGLES**][ **PIECE\_HEIGHT**][ **PIECE\_WIDTH**]

### 3.8.1 Field Documentation

#### 3.8.1.1 angle

```
size_t angle
```

### 3.8.1.2 id

size\_t id

### 3.8.1.3 shapes

int shapes[ **PIECE\_ANGLES**][ **PIECE\_HEIGHT**][ **PIECE\_WIDTH**]

### 3.8.1.4 x

size\_t x

### 3.8.1.5 y

size\_t y

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

- src/core/ **piece.h**



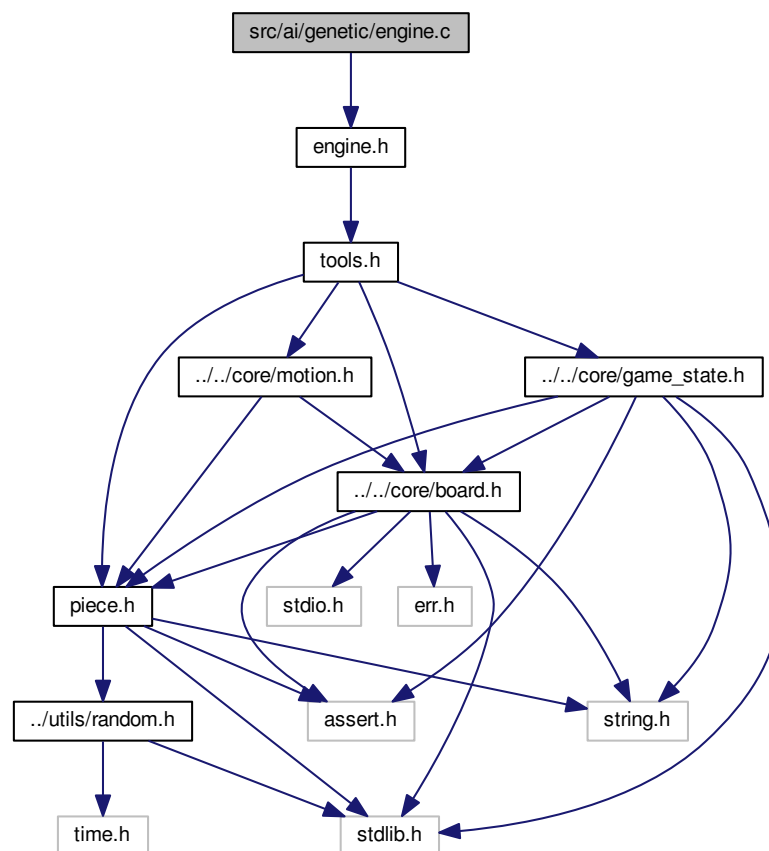
## Chapter 4

# File Documentation

### 4.1 src/ai/genetic/engine.c File Reference

```
#include "engine.h"
```

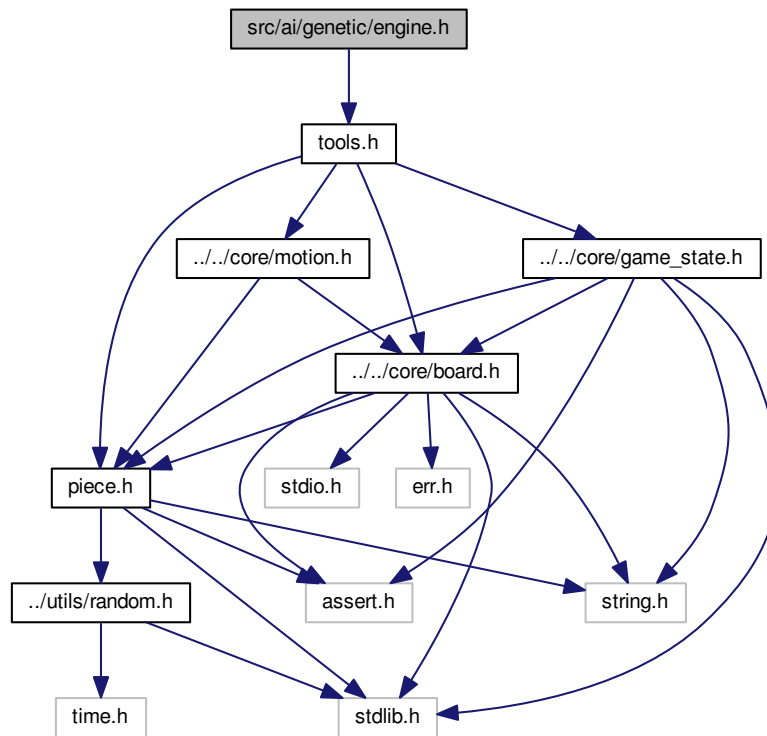
Include dependency graph for engine.c:



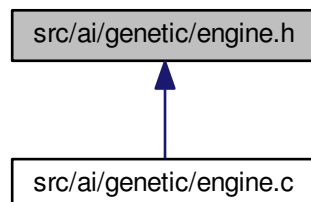
## 4.2 src/ai/genetic/engine.h File Reference

```
#include "tools.h"
```

Include dependency graph for engine.h:



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



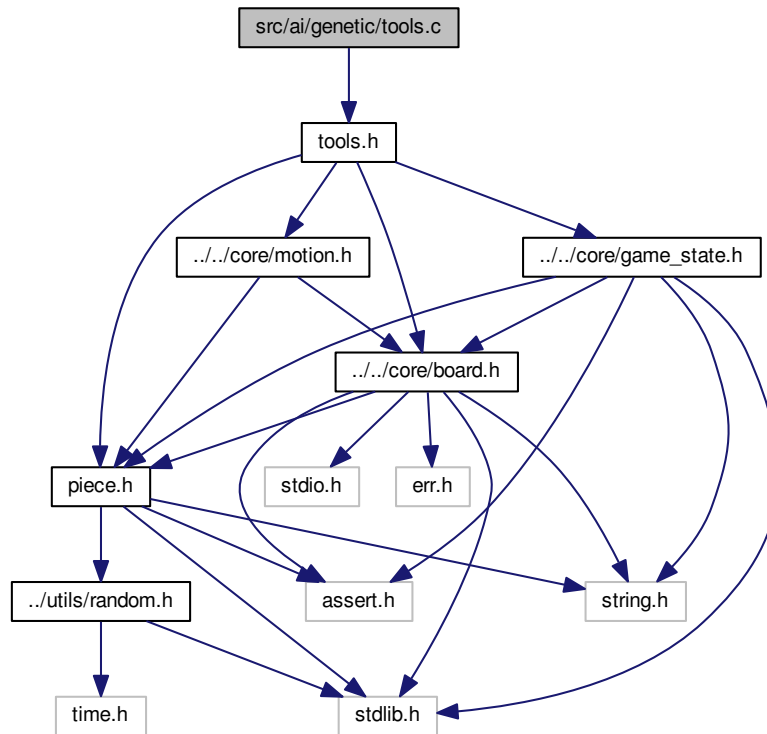
## Data Structures

- struct **ai\_coefs**



## 4.3 src/ai/genetic/tools.c File Reference

#include "tools.h"  
 Include dependency graph for tools.c:



### Functions

- void **board\_heights** (const struct **board** \*brd, size\_t \*heights)
- size\_t **board\_height** (const struct **board** \*brd, size\_t x)
- size\_t **bumpiness** (const struct **board** \*brd)
- size\_t **aggregate\_height** (const struct **board** \*brd)
- size\_t **hole** (const struct **board** \*brd, size\_t x)
- size\_t **holes** (const struct **board** \*brd)
- void **make\_line** (struct **board** \*brd, size\_t y)
- size\_t **clears** (const struct **board** \*brd)

### 4.3.1 Function Documentation

#### 4.3.1.1 aggregate\_height()

```
size_t aggregate_height (
    const struct board * brd ) [inline]
```

#### 4.3.1.2 board\_height()

```
size_t board_height (
    const struct board * brd,
    size_t x ) [inline]
```

#### 4.3.1.3 board\_heights()

```
void board_heights (
    const struct board * brd,
    size_t * heights ) [inline]
```

#### 4.3.1.4 bumpiness()

```
size_t bumpiness (
    const struct board * brd ) [inline]
```

#### 4.3.1.5 clears()

```
size_t clears (
    const struct board * brd ) [inline]
```

#### 4.3.1.6 hole()

```
size_t hole (
    const struct board * brd,
    size_t x ) [inline]
```

#### 4.3.1.7 holes()

```
size_t holes (
    const struct board * brd ) [inline]
```

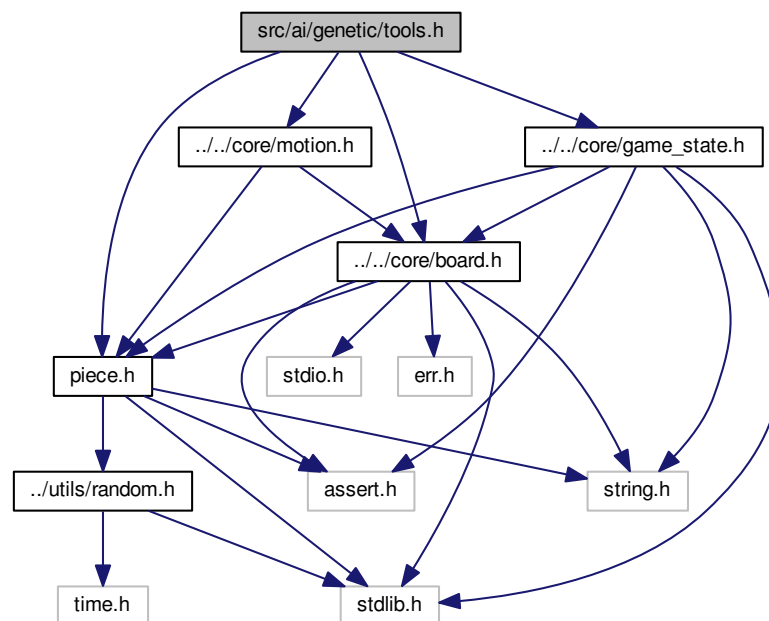
## 4.3.1.8 make\_line()

```
void make_line (  
    struct board * brd,  
    size_t y ) [inline]
```

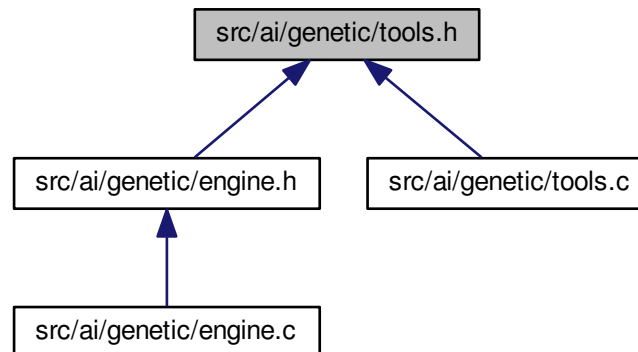
## 4.4 src/ai/genetic/tools.h File Reference

```
#include "../core/board.h"  
#include "../core/game_state.h"  
#include "../core/motion.h"  
#include "../core/piece.h"
```

Include dependency graph for tools.h:



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



## Macros

- `#define ABS(X) (((X) < 0) ? (-1 * (X)) : (X))`

## Functions

- `size_t board_height (const struct board *brd, size_t x)`
- `void board_heights (const struct board *brd, size_t *heights)`
- `size_t bumpiness (const struct board *brd)`
- `size_t aggregate_height (const struct board *brd)`
- `size_t hole (const struct board *brd, size_t x)`
- `size_t holes (const struct board *brd)`
- `size_t coalescent_clears (const struct board *brd)`
- `void make_line (struct board *brd, size_t y)`
- `size_t clears (const struct board *brd)`

### 4.4.1 Macro Definition Documentation

#### 4.4.1.1 ABS

```
#define ABS(  
    X ) (((X) < 0) ?  (-1 * (X)) : (X))
```

### 4.4.2 Function Documentation

#### 4.4.2.1 aggregate\_height()

```
size_t aggregate_height (
    const struct board * brd ) [inline]
```

#### 4.4.2.2 board\_height()

```
size_t board_height (
    const struct board * brd,
    size_t x ) [inline]
```

#### 4.4.2.3 board\_heights()

```
void board_heights (
    const struct board * brd,
    size_t * heights ) [inline]
```

#### 4.4.2.4 bumpiness()

```
size_t bumpiness (
    const struct board * brd ) [inline]
```

#### 4.4.2.5 clears()

```
size_t clears (
    const struct board * brd ) [inline]
```

#### 4.4.2.6 coalescent\_clears()

```
size_t coalescent_clears (
    const struct board * brd ) [inline]
```

#### 4.4.2.7 hole()

```
size_t hole (  
    const struct board * brd,  
    size_t x ) [inline]
```

#### 4.4.2.8 holes()

```
size_t holes (  
    const struct board * brd ) [inline]
```

#### 4.4.2.9 make\_line()

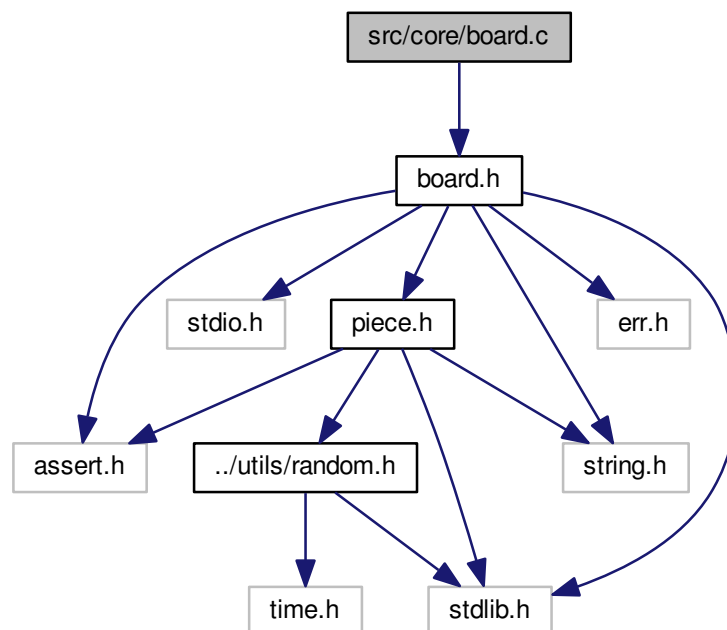
```
void make_line (  
    struct board * brd,  
    size_t y ) [inline]
```

### 4.5 src/core/board.c File Reference

No description.

```
#include "board.h"
```

Include dependency graph for board.c:



## Functions

- struct **board** \* **board\_create** ()
- void **board\_init** (struct **board** \*brd)
- void **board\_free** (struct **board** \*brd)
- struct **board** \* **board\_copy** (const struct **board** \*brd)
- int **board\_at** (const struct **board** \*brd, size\_t x, size\_t y)
- void **board\_set** (struct **board** \*brd, size\_t x, size\_t y, int state)
- void **board\_remove\_line** (struct **board** \*brd, size\_t line)
- void **board\_move\_line** (struct **board** \*brd, size\_t src, size\_t dest)
- int **board\_is\_line\_complete** (const struct **board** \*brd, size\_t line)
- size\_t **board\_get\_completed\_lines** (const struct **board** \*brd, size\_t \*\*lines)
- void **board\_break\_line** (struct **board** \*brd, size\_t line)
- void **board\_break\_lines** (struct **board** \*brd, const size\_t \*lines)
- void **board\_merge\_piece** (struct **board** \*brd, struct **piece** pc)
- int **board\_check\_position** (const struct **board** \*brd, struct **piece** pc)
- void **board\_print** (const struct **board** \*brd)

### 4.5.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.5.2 Function Documentation

#### 4.5.2.1 board\_at()

```
int board_at (  
    const struct board * brd,  
    size_t x,  
    size_t y ) [inline]
```

#### 4.5.2.2 board\_break\_line()

```
void board_break_line (  
    struct board * brd,  
    size_t line ) [inline]
```

#### 4.5.2.3 board\_break\_lines()

```
void board_break_lines (
    struct board * brd,
    const size_t * lines ) [inline]
```

#### 4.5.2.4 board\_check\_position()

```
int board_check_position (
    const struct board * brd,
    struct piece pc ) [inline]
```

#### 4.5.2.5 board\_copy()

```
struct board* board_copy (
    const struct board * brd )
```

#### 4.5.2.6 board\_create()

```
struct board* board_create ( )
```

#### 4.5.2.7 board\_free()

```
void board_free (
    struct board * brd ) [inline]
```

#### 4.5.2.8 board\_get\_completed\_lines()

```
size_t board_get_completed_lines (
    const struct board * brd,
    size_t ** lines ) [inline]
```

#### 4.5.2.9 board\_init()

```
void board_init (
    struct board * brd ) [inline]
```



#### 4.5.2.10 board\_is\_line\_complete()

```
int board_is_line_complete (
    const struct board * brd,
    size_t line ) [inline]
```

#### 4.5.2.11 board\_merge\_piece()

```
void board_merge_piece (
    struct board * brd,
    struct piece pc ) [inline]
```

#### 4.5.2.12 board\_move\_line()

```
void board_move_line (
    struct board * brd,
    size_t src,
    size_t dest ) [inline]
```

#### 4.5.2.13 board\_print()

```
void board_print (
    const struct board * brd ) [inline]
```

#### 4.5.2.14 board\_remove\_line()

```
void board_remove_line (
    struct board * brd,
    size_t line ) [inline]
```

#### 4.5.2.15 board\_set()

```
void board_set (
    struct board * brd,
    size_t x,
    size_t y,
    int state ) [inline]
```



## Macros

- `#define BOARD_WIDTH 10`
- `#define BOARD_HEIGHT 22`
- `#define BOARD_CELL_EMPTY (-1)`

## Functions

- `struct board * board_create ()`
- `void board_init (struct board *brd)`
- `void board_free (struct board *brd)`
- `struct board * board_copy (const struct board *brd)`
- `int board_at (const struct board *brd, size_t x, size_t y)`
- `void board_set (struct board *brd, size_t x, size_t y, int state)`
- `void board_remove_line (struct board *brd, size_t line)`
- `void board_move_line (struct board *brd, size_t src, size_t dest)`
- `int board_is_line_complete (const struct board *brd, size_t line)`
- `size_t board_get_completed_lines (const struct board *brd, size_t **lines)`
- `void board_break_line (struct board *brd, size_t line)`
- `void board_break_lines (struct board *brd, const size_t *lines)`
- `void board_merge_piece (struct board *brd, struct piece pc)`
- `int board_check_position (const struct board *brd, struct piece pc)`
- `void board_print (const struct board *brd)`

### 4.6.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 BOARD\_CELL\_EMPTY

```
#define BOARD_CELL_EMPTY (-1)
```

#### 4.6.2.2 BOARD\_HEIGHT

```
#define BOARD_HEIGHT 22
```

#### 4.6.2.3 BOARD\_WIDTH

```
#define BOARD_WIDTH 10
```

### 4.6.3 Function Documentation

#### 4.6.3.1 board\_at()

```
int board_at (
    const struct board * brd,
    size_t x,
    size_t y ) [inline]
```

#### 4.6.3.2 board\_break\_line()

```
void board_break_line (
    struct board * brd,
    size_t line ) [inline]
```

#### 4.6.3.3 board\_break\_lines()

```
void board_break_lines (
    struct board * brd,
    const size_t * lines ) [inline]
```

#### 4.6.3.4 board\_check\_position()

```
int board_check_position (
    const struct board * brd,
    struct piece pc ) [inline]
```

#### 4.6.3.5 board\_copy()

```
struct board* board_copy (
    const struct board * brd )
```

#### 4.6.3.6 board\_create()

```
struct board* board_create ( )
```

#### 4.6.3.7 board\_free()

```
void board_free (
    struct board * brd ) [inline]
```

#### 4.6.3.8 board\_get\_completed\_lines()

```
size_t board_get_completed_lines (
    const struct board * brd,
    size_t ** lines ) [inline]
```

#### 4.6.3.9 board\_init()

```
void board_init (
    struct board * brd ) [inline]
```

#### 4.6.3.10 board\_is\_line\_complete()

```
int board_is_line_complete (
    const struct board * brd,
    size_t line ) [inline]
```

#### 4.6.3.11 board\_merge\_piece()

```
void board_merge_piece (
    struct board * brd,
    struct piece pc ) [inline]
```

#### 4.6.3.12 board\_move\_line()

```
void board_move_line (
    struct board * brd,
    size_t src,
    size_t dest ) [inline]
```

#### 4.6.3.13 board\_print()

```
void board_print (
    const struct board * brd ) [inline]
```

#### 4.6.3.14 board\_remove\_line()

```
void board_remove_line (
    struct board * brd,
    size_t line ) [inline]
```

#### 4.6.3.15 board\_set()

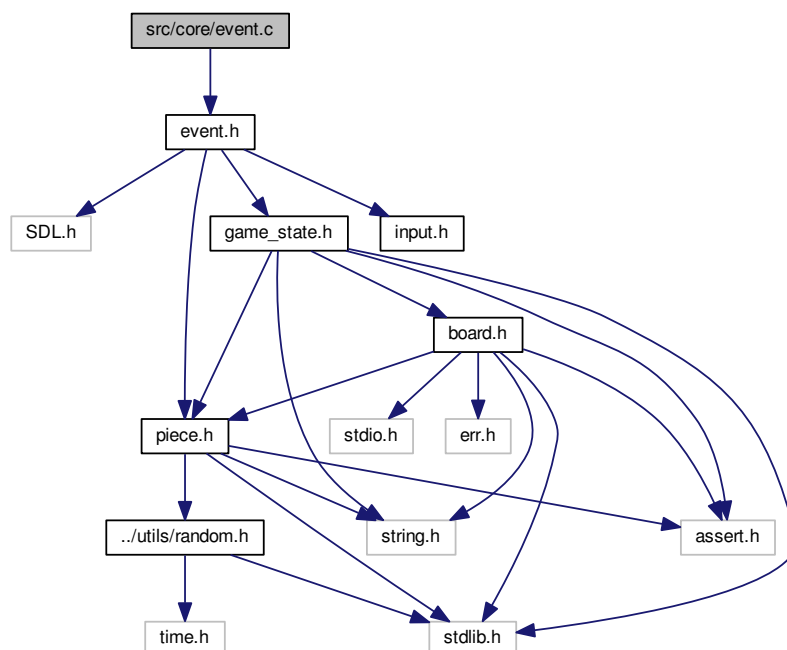
```
void board_set (
    struct board * brd,
    size_t x,
    size_t y,
    int state ) [inline]
```

## 4.7 src/core/event.c File Reference

No description.

```
#include "event.h"
```

Include dependency graph for event.c:



## Functions

- void **event\_handle** (struct **game\_state** \*gs, struct **input** \*in)

### 4.7.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.7.2 Function Documentation

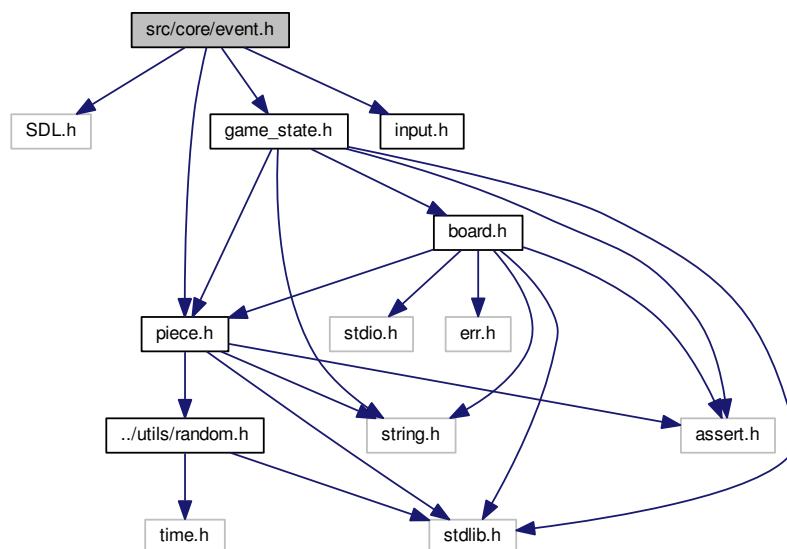
#### 4.7.2.1 event\_handle()

```
void event_handle (
    struct game_state * gs,
    struct input * in )
```

## 4.8 src/core/event.h File Reference

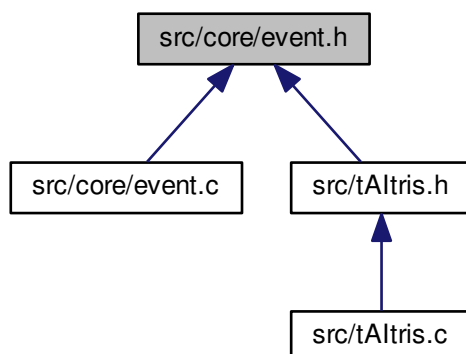
No description.

```
#include <SDL.h>
#include "game_state.h"
#include "input.h"
#include "piece.h"
Include dependency graph for event.h:
```





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



## Functions

- void **event\_handle** (struct **game\_state** \*gs, struct **input** \*in)

### 4.8.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.8.2 Function Documentation

#### 4.8.2.1 event\_handle()

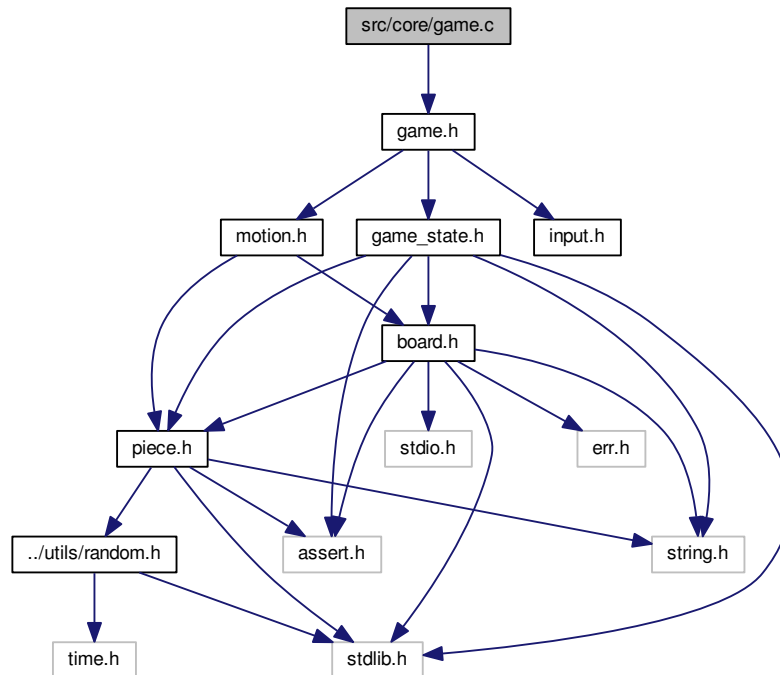
```
void event_handle (  
    struct game_state * gs,  
    struct input * in )
```

## 4.9 src/core/game.c File Reference

No description.

```
#include "game.h"
```

Include dependency graph for game.c:



### Functions

- void **game\_tick** (double dt, struct **game\_state** \*gs, struct **input** \*in)

### 4.9.1 Detailed Description

No description.

Author

S4MasterRace

Version

1.0

### 4.9.2 Function Documentation

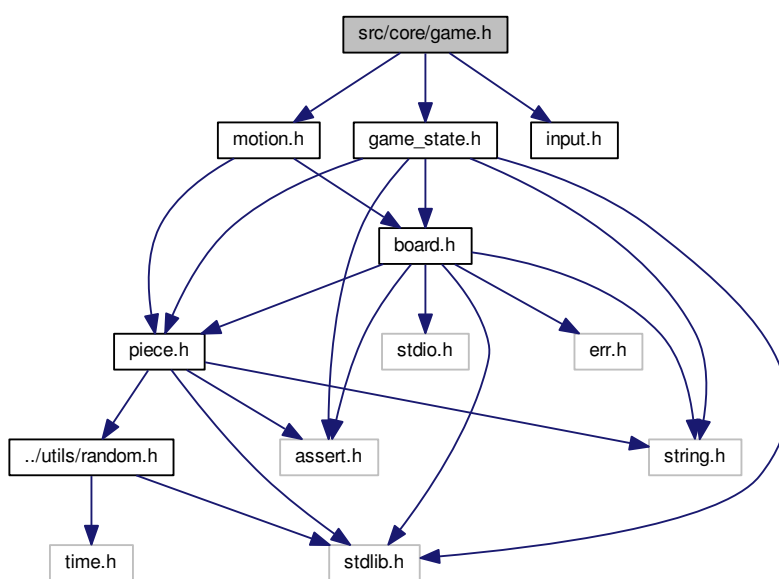
## 4.9.2.1 game\_tick()

```
void game_tick (
    double dt,
    struct game_state * gs,
    struct input * in )
```

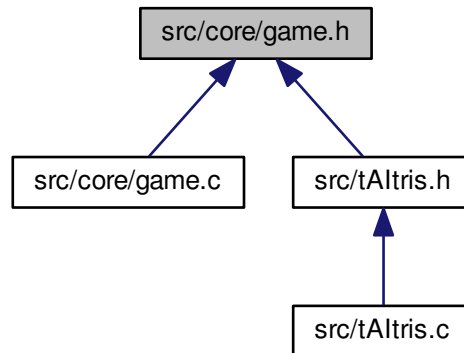
## 4.10 src/core/game.h File Reference

No description.

```
#include "game_state.h"
#include "motion.h"
#include "input.h"
Include dependency graph for game.h:
```



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



## Functions

- void **game\_tick** (double dt, struct **game\_state** \*gs, struct **input** \*in)

### 4.10.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.10.2 Function Documentation

#### 4.10.2.1 game\_tick()

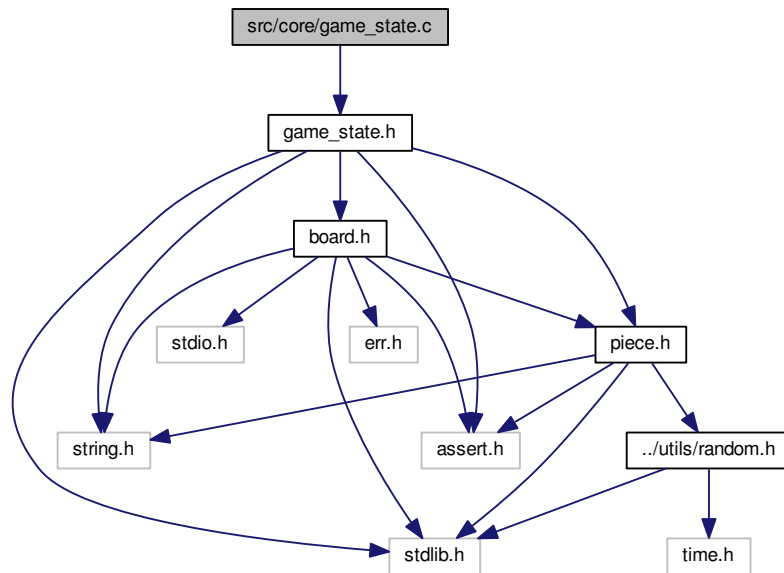
```
void game_tick (
    double dt,
    struct game_state * gs,
    struct input * in )
```

## 4.11 src/core/game\_state.c File Reference

No description.

```
#include "game_state.h"
```

Include dependency graph for game\_state.c:



### Functions

- struct **game\_state** \* **gs\_create** ()
- void **gs\_init** (struct **game\_state** \*gs)
- void **gs\_free** (struct **game\_state** \*gs)
- int **gs\_next\_piece** (struct **game\_state** \*gs)

#### 4.11.1 Detailed Description

No description.

Author

S4MasterRace

Version

1.0

#### 4.11.2 Function Documentation

#### 4.11.2.1 gs\_create()

```
struct game_state* gs_create ( )
```

#### 4.11.2.2 gs\_free()

```
void gs_free (
    struct game_state * gs ) [inline]
```

#### 4.11.2.3 gs\_init()

```
void gs_init (
    struct game_state * gs ) [inline]
```

#### 4.11.2.4 gs\_next\_piece()

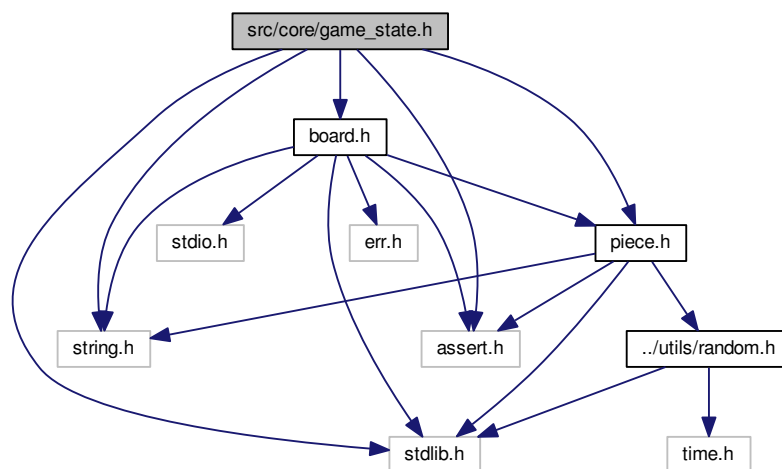
```
int gs_next_piece (
    struct game_state * gs ) [inline]
```

## 4.12 src/core/game\_state.h File Reference

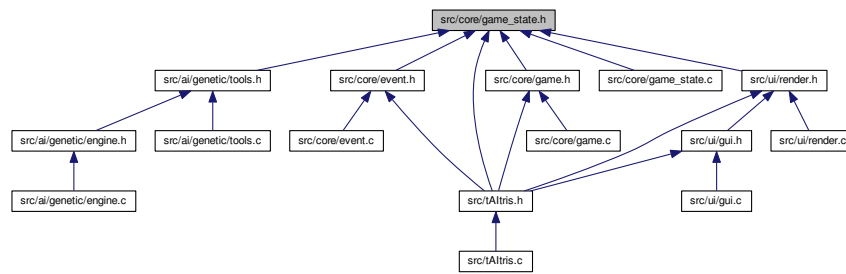
No description.

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "board.h"
#include "piece.h"
```

Include dependency graph for game\_state.h:



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



## Data Structures

- struct **game\_state**

## Macros

- #define **GS\_STATE\_PAUSED** 0
- #define **GS\_STATE\_PLAYING** 1
- #define **GS\_STATE\_GAMEOVER** 2
- #define **GS\_STATE\_QUIT** 3
- #define **GS\_SPAWN\_X** (**BOARD\_WIDTH** / 2 - **PIECE\_WIDTH** / 2)
- #define **GS\_SPAWN\_Y** 0

## Functions

- struct **game\_state** \* **gs\_create** ()
- void **gs\_init** (struct **game\_state** \*gs)
- void **gs\_free** (struct **game\_state** \*gs)
- int **gs\_next\_piece** (struct **game\_state** \*gs)

### 4.12.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.12.2 Macro Definition Documentation

#### 4.12.2.1 GS\_SPAWN\_X

```
#define GS_SPAWN_X ( BOARD_WIDTH / 2 - PIECE_WIDTH / 2)
```

#### 4.12.2.2 GS\_SPAWN\_Y

```
#define GS_SPAWN_Y 0
```

#### 4.12.2.3 GS\_STATE\_GAMEOVER

```
#define GS_STATE_GAMEOVER 2
```

#### 4.12.2.4 GS\_STATE\_PAUSED

```
#define GS_STATE_PAUSED 0
```

#### 4.12.2.5 GS\_STATE\_PLAYING

```
#define GS_STATE_PLAYING 1
```

#### 4.12.2.6 GS\_STATE\_QUIT

```
#define GS_STATE_QUIT 3
```

### 4.12.3 Function Documentation

#### 4.12.3.1 gs\_create()

```
struct game_state* gs_create ( )
```



#### 4.12.3.2 gs\_free()

```
void gs_free (
    struct game_state * gs ) [inline]
```

#### 4.12.3.3 gs\_init()

```
void gs_init (
    struct game_state * gs ) [inline]
```

#### 4.12.3.4 gs\_next\_piece()

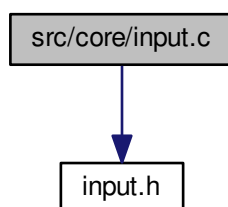
```
int gs_next_piece (
    struct game_state * gs ) [inline]
```

## 4.13 src/core/input.c File Reference

No description.

```
#include "input.h"
```

Include dependency graph for input.c:



### 4.13.1 Detailed Description

No description.

Author

S4MasterRace

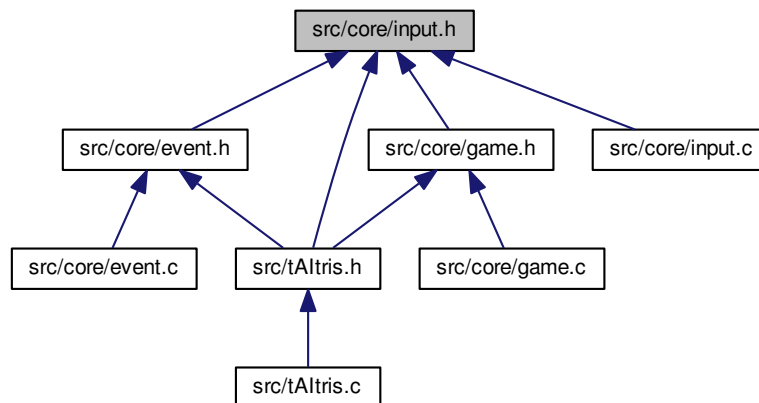
Version

1.0

## 4.14 src/core/input.h File Reference

No description.

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



### Data Structures

- struct **input**

#### 4.14.1 Detailed Description

No description.

Author

S4MasterRace

Version

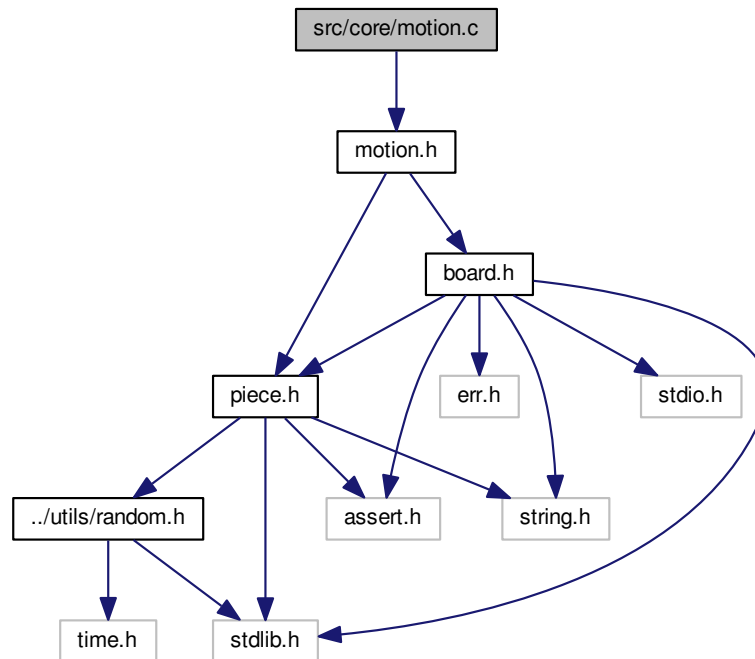
1.0

## 4.15 src/core/motion.c File Reference

No description.

```
#include "motion.h"
```

Include dependency graph for motion.c:



## Functions

- int **motion\_can\_move** (struct **piece** pc, const struct **board** \*brd, int dx, int dy)
- int **motion\_can\_rotate** (struct **piece** pc, const struct **board** \*brd, int rotation)
- int **motion\_try\_move** (struct **piece** \*pc, const struct **board** \*brd, int dx, int dy)
- int **motion\_try\_rotate** (struct **piece** \*pc, const struct **board** \*brd, int rotation)
- int **motion\_try\_move\_down** (struct **piece** \*pc, const struct **board** \*brd)

### 4.15.1 Detailed Description

No description.

Author

S4MasterRace

Version

1.0

### 4.15.2 Function Documentation

#### 4.15.2.1 motion\_can\_move()

```
int motion_can_move (
    struct piece pc,
    const struct board * brd,
    int dx,
    int dy )
```

#### 4.15.2.2 motion\_can\_rotate()

```
int motion_can_rotate (
    struct piece pc,
    const struct board * brd,
    int rotation )
```

#### 4.15.2.3 motion\_try\_move()

```
int motion_try_move (
    struct piece * pc,
    const struct board * brd,
    int dx,
    int dy )
```

#### 4.15.2.4 motion\_try\_move\_down()

```
int motion_try_move_down (
    struct piece * pc,
    const struct board * brd )
```

#### 4.15.2.5 motion\_try\_rotate()

```
int motion_try_rotate (
    struct piece * pc,
    const struct board * brd,
    int rotation )
```

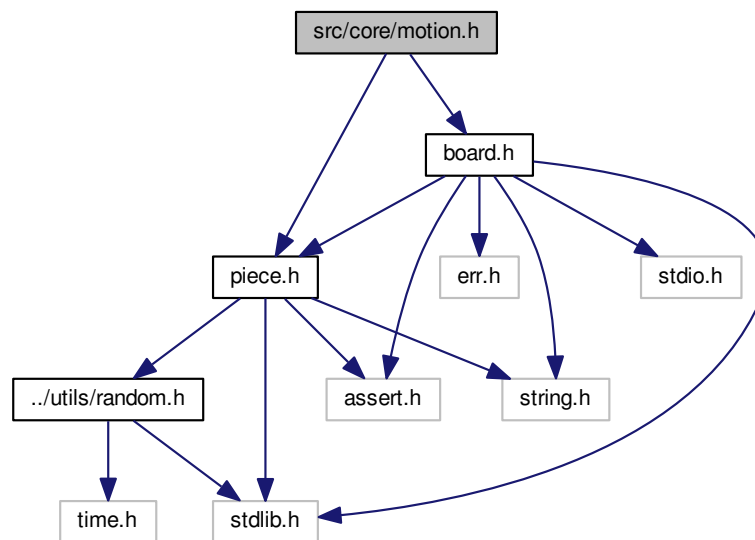
## 4.16 src/core/motion.h File Reference

No description.

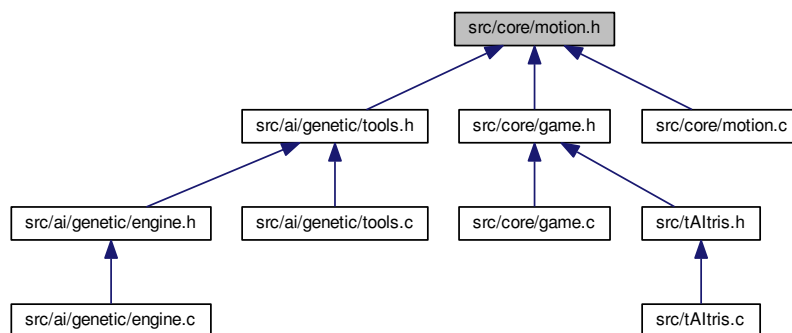
```
#include "board.h"
```

```
#include "piece.h"
```

Include dependency graph for motion.h:



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



### Functions

- int **motion\_can\_move** (struct **piece** pc, const struct **board** \*brd, int dx, int dy)
- int **motion\_can\_rotate** (struct **piece** pc, const struct **board** \*brd, int rotation)
- int **motion\_try\_move** (struct **piece** \*pc, const struct **board** \*brd, int dx, int dy)
- int **motion\_try\_rotate** (struct **piece** \*pc, const struct **board** \*brd, int rotation)
- int **motion\_try\_move\_down** (struct **piece** \*pc, const struct **board** \*brd)

### 4.16.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.16.2 Function Documentation

#### 4.16.2.1 motion\_can\_move()

```
int motion_can_move (
    struct piece pc,
    const struct board * brd,
    int dx,
    int dy )
```

#### 4.16.2.2 motion\_can\_rotate()

```
int motion_can_rotate (
    struct piece pc,
    const struct board * brd,
    int rotation )
```

#### 4.16.2.3 motion\_try\_move()

```
int motion_try_move (
    struct piece * pc,
    const struct board * brd,
    int dx,
    int dy )
```

## 4.16.2.4 motion\_try\_move\_down()

```
int motion_try_move_down (
    struct piece * pc,
    const struct board * brd )
```

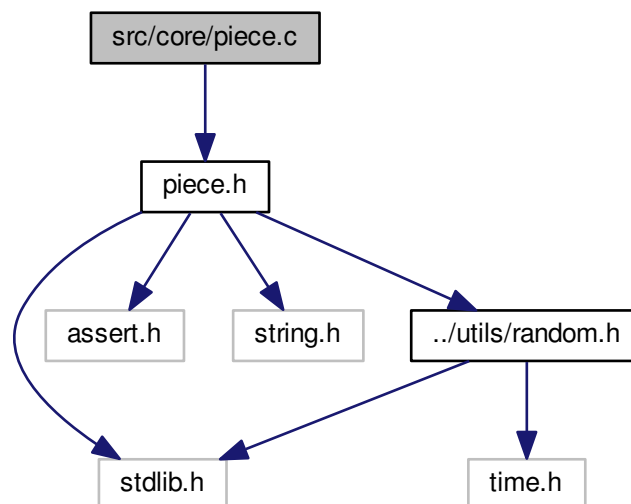
## 4.16.2.5 motion\_try\_rotate()

```
int motion_try_rotate (
    struct piece * pc,
    const struct board * brd,
    int rotation )
```

## 4.17 src/core/piece.c File Reference

No description.

```
#include "piece.h"
Include dependency graph for piece.c:
```



## Functions

- void **piece\_random** (struct **piece** \**pc*, size\_t *x*, size\_t *y*)
- void **piece\_move** (struct **piece** \**pc*, int *dx*, int *dy*)
- void **piece\_rotate** (struct **piece** \**pc*, int *rotation*)

## Variables

- const int **PIECE\_SHAPES** [7][4][4][4]

### 4.17.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.17.2 Function Documentation

#### 4.17.2.1 piece\_move()

```
void piece_move (
    struct piece * pc,
    int dx,
    int dy ) [inline]
```

#### 4.17.2.2 piece\_random()

```
void piece_random (
    struct piece * pc,
    size_t x,
    size_t y ) [inline]
```

#### 4.17.2.3 piece\_rotate()

```
void piece_rotate (
    struct piece * pc,
    int rotation ) [inline]
```

### 4.17.3 Variable Documentation



## 4.17.3.1 PIECE\_SHAPES

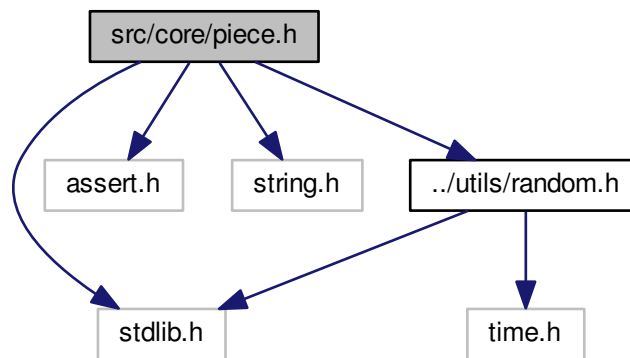
```
const int PIECE_SHAPES[7][4][4][4]
```

## 4.18 src/core/piece.h File Reference

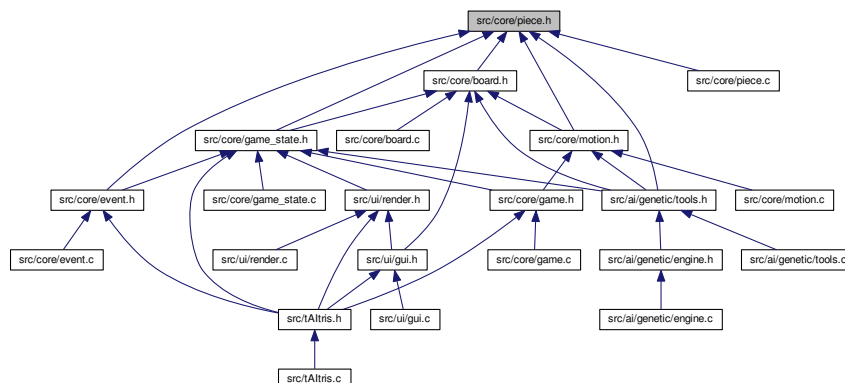
No description.

```
#include <stdlib.h>
#include <assert.h>
#include <string.h>
#include "../utils/random.h"
```

Include dependency graph for piece.h:



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



## Data Structures

- struct **piece**

## Macros

- `#define` **PIECE\_I** 0
- `#define` **PIECE\_O** 1
- `#define` **PIECE\_T** 2
- `#define` **PIECE\_L** 3
- `#define` **PIECE\_J** 4
- `#define` **PIECE\_Z** 5
- `#define` **PIECE\_S** 6
- `#define` **PIECE\_COUNT** 7
- `#define` **PIECE\_WIDTH** 4
- `#define` **PIECE\_HEIGHT** 4
- `#define` **PIECE\_ANGLE\_UP** 0
- `#define` **PIECE\_ANGLE\_RIGHT** 1
- `#define` **PIECE\_ANGLE\_DOWN** 2
- `#define` **PIECE\_ANGLE\_LEFT** 3
- `#define` **PIECE\_ANGLES** 4
- `#define` **PIECE\_ROTATE\_LEFT** (-1)
- `#define` **PIECE\_ROTATE\_RIGHT** 1

## Functions

- void **piece\_random** (struct **piece** \*pc, size\_t x, size\_t y)
- void **piece\_move** (struct **piece** \*pc, int dx, int dy)
- void **piece\_rotate** (struct **piece** \*pc, int rotation)

## Variables

- const int **PIECE\_SHAPES** [ **PIECE\_COUNT**][ **PIECE\_ANGLES**][ **PIECE\_HEIGHT**][ **PIECE\_WIDTH**]

### 4.18.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.18.2 Macro Definition Documentation

#### 4.18.2.1 PIECE\_ANGLE\_DOWN

```
#define PIECE_ANGLE_DOWN 2
```

#### 4.18.2.2 PIECE\_ANGLE\_LEFT

```
#define PIECE_ANGLE_LEFT 3
```

#### 4.18.2.3 PIECE\_ANGLE\_RIGHT

```
#define PIECE_ANGLE_RIGHT 1
```

#### 4.18.2.4 PIECE\_ANGLE\_UP

```
#define PIECE_ANGLE_UP 0
```

#### 4.18.2.5 PIECE\_ANGLES

```
#define PIECE_ANGLES 4
```

#### 4.18.2.6 PIECE\_COUNT

```
#define PIECE_COUNT 7
```

#### 4.18.2.7 PIECE\_HEIGHT

```
#define PIECE_HEIGHT 4
```

#### 4.18.2.8 PIECE\_I

```
#define PIECE_I 0
```

#### 4.18.2.9 PIECE\_J

```
#define PIECE_J 4
```

#### 4.18.2.10 PIECE\_L

```
#define PIECE_L 3
```

#### 4.18.2.11 PIECE\_O

```
#define PIECE_O 1
```

#### 4.18.2.12 PIECE\_ROTATE\_LEFT

```
#define PIECE_ROTATE_LEFT (-1)
```

#### 4.18.2.13 PIECE\_ROTATE\_RIGHT

```
#define PIECE_ROTATE_RIGHT 1
```

#### 4.18.2.14 PIECE\_S

```
#define PIECE_S 6
```

#### 4.18.2.15 PIECE\_T

```
#define PIECE_T 2
```

#### 4.18.2.16 PIECE\_WIDTH

```
#define PIECE_WIDTH 4
```

#### 4.18.2.17 PIECE\_Z

```
#define PIECE_Z 5
```

### 4.18.3 Function Documentation

#### 4.18.3.1 piece\_move()

```
void piece_move (
    struct piece * pc,
    int dx,
    int dy ) [inline]
```

#### 4.18.3.2 piece\_random()

```
void piece_random (
    struct piece * pc,
    size_t x,
    size_t y ) [inline]
```

#### 4.18.3.3 piece\_rotate()

```
void piece_rotate (
    struct piece * pc,
    int rotation ) [inline]
```

### 4.18.4 Variable Documentation

#### 4.18.4.1 PIECE\_SHAPES

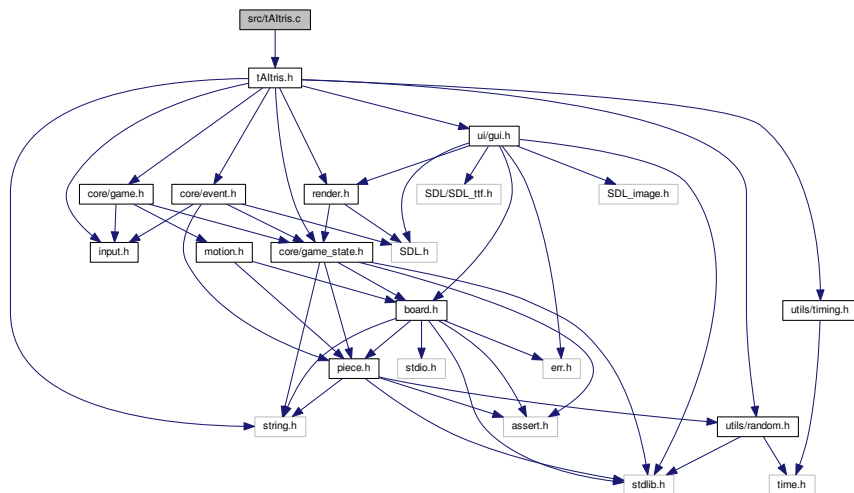
```
const int PIECE_SHAPES[ PIECE_COUNT][ PIECE_ANGLES][ PIECE_HEIGHT][ PIECE_WIDTH]
```

## 4.19 src/tAltris.c File Reference

Main file.

```
#include "tAltris.h"
```

Include dependency graph for tAltris.c:



### Functions

- int **main** ()

### 4.19.1 Detailed Description

Main file.

Author

S4MasterRace

Version

1.0

### 4.19.2 Function Documentation

#### 4.19.2.1 main()

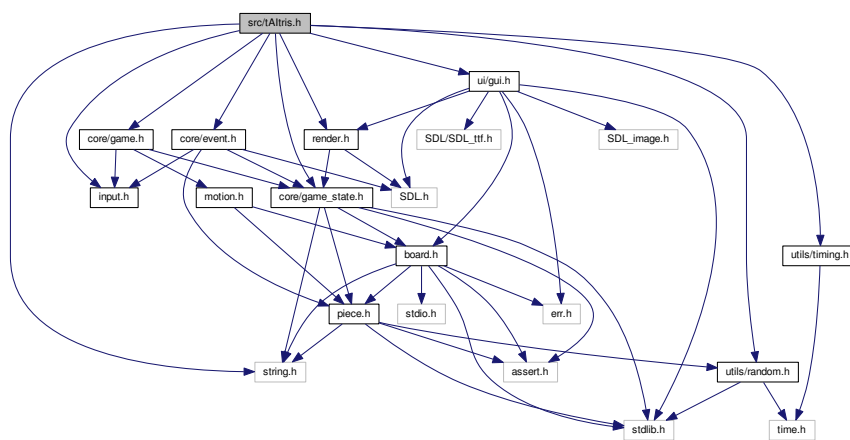
```
int main ( )
```

## 4.20 src/tAltris.h File Reference

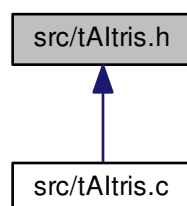
Main file.

```
#include <string.h>
#include "utils/random.h"
#include "utils/timing.h"
#include "core/game_state.h"
#include "core/event.h"
#include "core/game.h"
#include "core/input.h"
#include "ui/gui.h"
#include "ui/render.h"
```

Include dependency graph for tAltris.h:



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



### 4.20.1 Detailed Description

Main file.

Author

S4MasterRace

Version

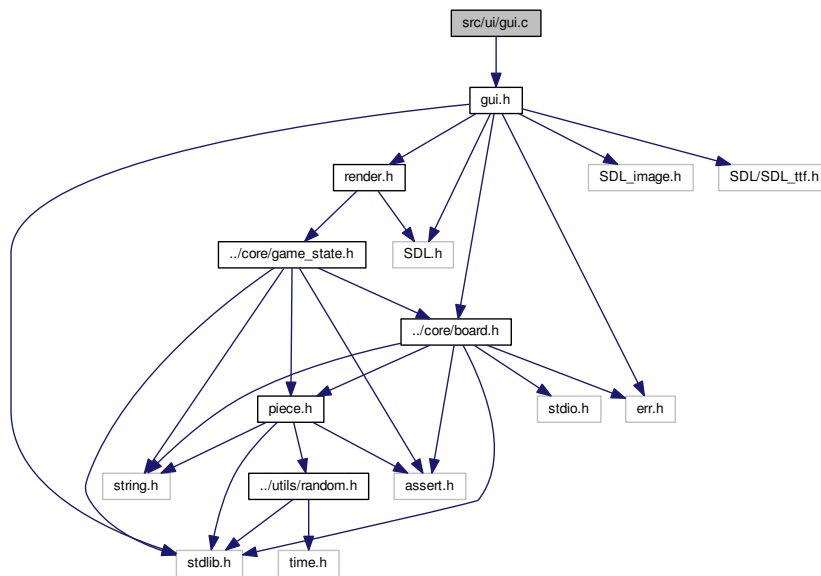
1.0

## 4.21 src/ui/gui.c File Reference

No description.

```
#include "gui.h"
```

Include dependency graph for gui.c:



### Functions

- `SDL_Surface *` **gui\_init** ()
- `void` **gui\_free** (SDL\_Surface \*win)
- `SDL_Surface *` **gui\_load\_image** (char \*path)

#### 4.21.1 Detailed Description

No description.

Author

S4MasterRace

Version

1.0



## 4.21.2 Function Documentation

### 4.21.2.1 gui\_free()

```
void gui_free (
    SDL_Surface * win ) [inline]
```

### 4.21.2.2 gui\_init()

```
SDL_Surface* gui_init ( ) [inline]
```

### 4.21.2.3 gui\_load\_image()

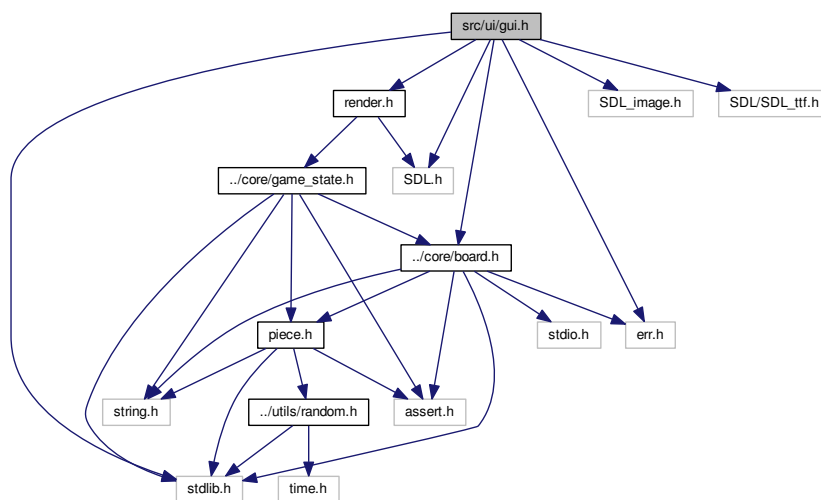
```
SDL_Surface* gui_load_image (
    char * path ) [inline]
```

## 4.22 src/ui/gui.h File Reference

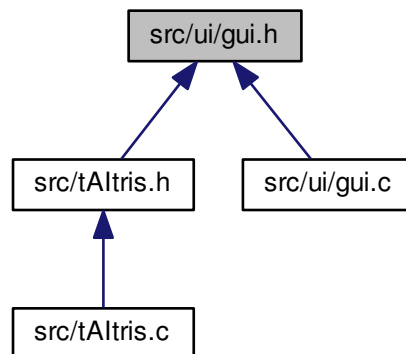
No description.

```
#include <stdlib.h>
#include <err.h>
#include <SDL.h>
#include <SDL_image.h>
#include <SDL/SDL_ttf.h>
#include "../core/board.h"
#include "render.h"
```

Include dependency graph for gui.h:



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



## Macros

- `#define GUI_TITLE "tAltris"`
- `#define GUI_WIDTH ( BOARD_WIDTH * RENDER_CELL_SIZE + 500)`
- `#define GUI_HEIGHT ( BOARD_HEIGHT * RENDER_CELL_SIZE)`

## Functions

- `SDL_Surface * gui_init ()`
- `void gui_free (SDL_Surface *win)`
- `SDL_Surface * gui_load_image (char *path)`

### 4.22.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.22.2 Macro Definition Documentation

#### 4.22.2.1 GUI\_HEIGHT

```
#define GUI_HEIGHT ( BOARD_HEIGHT * RENDER_CELL_SIZE)
```

#### 4.22.2.2 GUI\_TITLE

```
#define GUI_TITLE "tAItiris"
```

#### 4.22.2.3 GUI\_WIDTH

```
#define GUI_WIDTH ( BOARD_WIDTH * RENDER_CELL_SIZE + 500)
```

### 4.22.3 Function Documentation

#### 4.22.3.1 gui\_free()

```
void gui_free (
    SDL_Surface * win ) [inline]
```

#### 4.22.3.2 gui\_init()

```
SDL_Surface* gui_init ( ) [inline]
```

#### 4.22.3.3 gui\_load\_image()

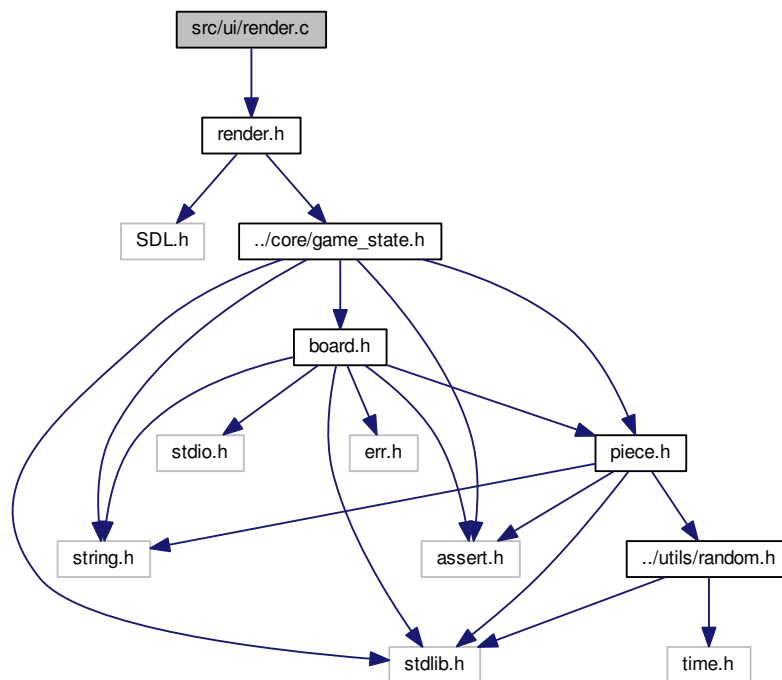
```
SDL_Surface* gui_load_image (
    char * path ) [inline]
```

## 4.23 src/ui/render.c File Reference

No description.

```
#include "render.h"
```

Include dependency graph for render.c:



### Functions

- void **render\_handle** (SDL\_Surface \*screen, const struct **game\_state** \*gs)
- void **render\_board** (SDL\_Surface \*screen, const struct **board** \*brd)
- void **render\_piece** (SDL\_Surface \*screen, struct **piece** pc)
- void **render\_next\_piece** (SDL\_Surface \*screen, struct **piece** pc)

### 4.23.1 Detailed Description

No description.

Author

S4MasterRace

Version

1.0

## 4.23.2 Function Documentation

### 4.23.2.1 render\_board()

```
void render_board (
    SDL_Surface * screen,
    const struct board * brd )
```

### 4.23.2.2 render\_handle()

```
void render_handle (
    SDL_Surface * screen,
    const struct game_state * gs )
```

### 4.23.2.3 render\_next\_piece()

```
void render_next_piece (
    SDL_Surface * screen,
    struct piece pc )
```

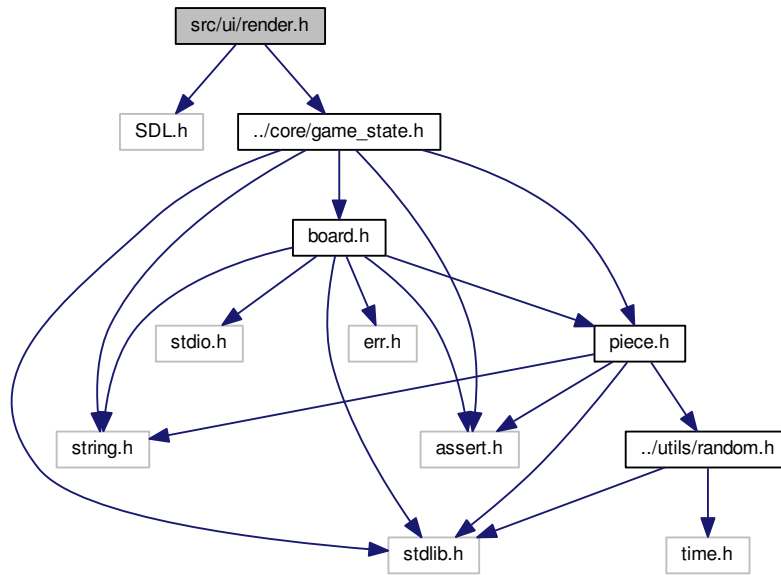
### 4.23.2.4 render\_piece()

```
void render_piece (
    SDL_Surface * screen,
    struct piece pc )
```

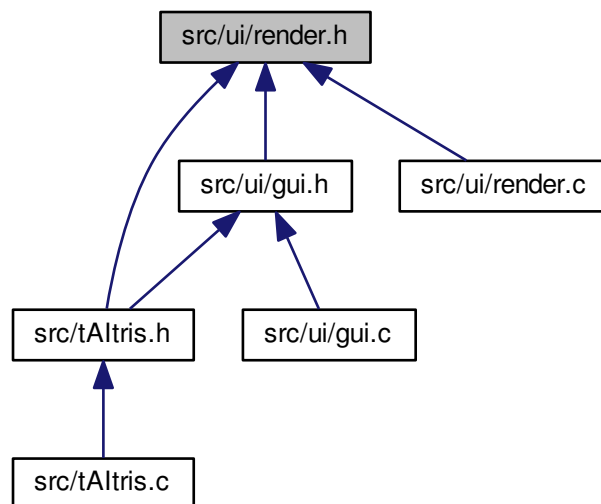
## 4.24 src/ui/render.h File Reference

No description.

```
#include <SDL.h>
#include "../core/game_state.h"
Include dependency graph for render.h:
```



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



## Macros

- `#define RENDER_FPS 30`

- `#define RENDER_CELL_SIZE 64`

## Functions

- void **render\_handle** (SDL\_Surface \*screen, const struct **game\_state** \*gs)
- void **render\_board** (SDL\_Surface \*screen, const struct **board** \*brd)
- void **render\_piece** (SDL\_Surface \*screen, struct **piece** pc)
- void **render\_next\_piece** (SDL\_Surface \*screen, struct **piece** pc)

### 4.24.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.24.2 Macro Definition Documentation

#### 4.24.2.1 RENDER\_CELL\_SIZE

```
#define RENDER_CELL_SIZE 64
```

#### 4.24.2.2 RENDER\_FPS

```
#define RENDER_FPS 30
```

### 4.24.3 Function Documentation

#### 4.24.3.1 render\_board()

```
void render_board (
    SDL_Surface * screen,
    const struct board * brd )
```

#### 4.24.3.2 render\_handle()

```
void render_handle (
    SDL_Surface * screen,
    const struct game_state * gs )
```

#### 4.24.3.3 render\_next\_piece()

```
void render_next_piece (
    SDL_Surface * screen,
    struct piece pc )
```

#### 4.24.3.4 render\_piece()

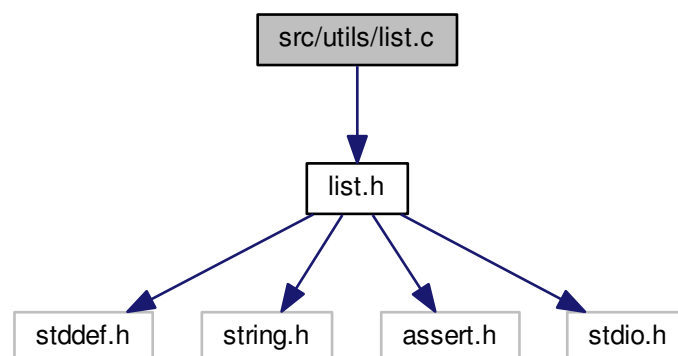
```
void render_piece (
    SDL_Surface * screen,
    struct piece pc )
```

## 4.25 src/utls/list.c File Reference

Intrusive list implement.

```
#include "list.h"
```

Include dependency graph for list.c:





## Functions

- void **list\_init** (struct **list** \* **list**)
- size\_t **list\_length** (const struct **list** \* **list**)
- struct **list\_node** \* **list\_first** (const struct **list** \* **list**)
- struct **list\_node** \* **list\_last** (const struct **list** \* **list**)
- struct **list\_node** \* **list\_next** (const struct **list\_node** \*node)
- struct **list\_node** \* **list\_advance** (struct **list\_node** \*node, size\_t distance)
- struct **list\_node** \* **list\_at** (const struct **list** \* **list**, size\_t pos)
- void **list\_reverse** (struct **list** \* **list**)
- void **list\_swap** (struct **list** \*l1, struct **list** \*l2)
- void **list\_split\_at** (struct **list** \* **list**, size\_t pos, struct **list** \*right)
- void **list\_concat** (struct **list** \*l1, struct **list** \*l2)
- void **list\_sort** (struct **list** \* **list**, int(\*cmp)(struct **list\_node** \*, struct **list\_node** \*))
- int **list\_is\_empty** (const struct **list** \* **list**)
- void **list\_add** (struct **list** \* **list**, struct **list\_node** \*node)
- void **list\_append** (struct **list** \* **list**, struct **list\_node** \*node)
- void **list\_insert\_after** (struct **list** \* **list**, struct **list\_node** \*curr, struct **list\_node** \*node)
- void **list\_insert\_at** (struct **list** \* **list**, struct **list\_node** \*node, size\_t pos)
- void **list\_del** (struct **list** \* **list**)
- void **list\_del\_after** (struct **list** \* **list**, struct **list\_node** \*node)
- void **list\_del\_at** (struct **list** \* **list**, size\_t pos)
- void **list\_print** (const struct **list** \* **list**)

### 4.25.1 Detailed Description

Intrusive list implement.

#### Author

S4MasterRace

#### Version

1.0

### 4.25.2 Function Documentation

#### 4.25.2.1 list\_add()

```
void list_add (
    struct list * list,
    struct list_node * node ) [inline]
```

Adds *node* in the front of *list*

**Parameters**

<i>list</i>	a list.
<i>node</i>	the new node.

**Precondition**

`list` must be not NULL.  
`node` must be not NULL.

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(1)

**4.25.2.2 list\_advance()**

```
struct list_node* list_advance (
    struct list_node * node,
    size_t distance )
```

Returns the *nth*-node after the current one.

**Parameters**

<i>node</i>	a node.
<i>distance</i>	distance to move on.

**Returns**

the *nth*-node after `node`.

**Precondition**

`node` must be not NULL.

**Remarks**

Complexity: O(*n*)

#### 4.25.2.3 list\_append()

```
void list_append (
    struct list * list,
    struct list_node * node ) [inline]
```

Adds *node* at the end of *list*.

**Parameters**

<i>list</i>	a list.
<i>node</i>	the new node.

**Precondition**

`list` must be not NULL.  
`node` must be not NULL.

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(n)

**4.25.2.4 list\_at()**

```
struct list_node* list_at (  
    const struct list * list,  
    size_t pos )
```

Returns node at the position `pos`.

**Parameters**

<i>list</i>	a list.
<i>pos</i>	position (0-based) of the node.

**Returns**

the node at the position `pos`.

**Precondition**

`list` must be not NULL.  
`list` must be not empty.  
`pos` must be in `[0; list_length(list)[`.

**Remarks**

Complexity: O(N)

#### 4.25.2.5 list\_concat()

```
void list_concat (
    struct list * l1,
    struct list * l2 ) [inline]
```

Concatenates two lists.

##### Parameters

<i>l1</i>	list 1.
<i>l2</i>	list 2.

##### Precondition

*l1* must be not NULL.  
*l2* must be not NULL.  
*l1* must be different of *l2*.

##### Postcondition

*l2* is reset to an empty list.

##### Remarks

Complexity: O(N)

#### 4.25.2.6 list\_del()

```
void list_del (
    struct list * list ) [inline]
```

Deletes the first node.

##### Parameters

<i>list</i>	a list.
-------------	---------

##### Precondition

*list* must be not NULL.  
*list* must be not empty.

##### Postcondition

List size decreases by 1.

**Remarks**

Complexity: O(1)

**4.25.2.7 list\_del\_after()**

```
void list_del_after (
    struct list * list,
    struct list_node * node ) [inline]
```

Deletes the node at after the node `curr`.

**Parameters**

<i>list</i>	a list.
<i>node</i>	a node of list.

**Precondition**

`list` must be not NULL.  
`node` must be not NULL.  
`list` must be not empty.  
`node` must a node of `list`.

**Postcondition**

List size decreases by 1.

**Remarks**

Complexity: O(1)

**4.25.2.8 list\_del\_at()**

```
void list_del_at (
    struct list * list,
    size_t pos ) [inline]
```

Deletes the node at the position `pos`.

**Parameters**

<i>list</i>	a list.
<i>pos</i>	index (0-based) of the node to delete.

**Precondition**

`list` must be not NULL.  
`list` must be not empty.  
`pos` must be in `[0; list_length(list)]`.

**Postcondition**

List size decreases by 1.

**Remarks**

Complexity:  $O(n)$

**4.25.2.9 list\_first()**

```
struct list_node* list_first (
    const struct list * list )
```

Returns the first node.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

the first node.

**Precondition**

`list` must be not NULL.  
`list` must be not empty.

**Remarks**

Complexity:  $O(1)$

**4.25.2.10 list\_init()**

```
void list_init (
    struct list * list ) [inline]
```

Initializes the list.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Precondition**

`list` must be not NULL.

**Postcondition**

`list` is empty.

`list` has a size of 0.

**Remarks**

Complexity: O(1)

**4.25.2.11 `list_insert_after()`**

```
void list_insert_after (  
    struct list * list,  
    struct list_node * curr,  
    struct list_node * node ) [inline]
```

Inserts `node` at after the node `curr`.

**Parameters**

<i>list</i>	a list.
<i>curr</i>	a node of <code>list</code> .
<i>node</i>	new node.

**Precondition**

`list` must be not NULL.

`curr` must be not NULL.

`curr` must a node of `list`.

`node` must be not NULL.

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(1)



## 4.25.2.12 list\_insert\_at()

```
void list_insert_at (
    struct list * list,
    struct list_node * node,
    size_t pos ) [inline]
```

Inserts *node* at the position *pos* in *list*.

## Parameters

<i>list</i>	a list.
<i>node</i>	new node.
<i>pos</i>	position (0-based) where to insert the new node.

## Precondition

*list* must be not NULL.  
*node* must be not NULL.  
*pos* must be in [0; list\_length(*list*)].

## Postcondition

List size increases by 1.

## Remarks

Complexity: O(n)

## 4.25.2.13 list\_is\_empty()

```
int list_is_empty (
    const struct list * list ) [inline]
```

Tests if a list is empty.

## Parameters

<i>list</i>	a list.
-------------	---------

## Returns

1 if the list is empty, otherwise 0.

## Precondition

*list* must be not NULL.

**Remarks**

Complexity: O(1)

**4.25.2.14 list\_last()**

```
struct list_node* list_last (
    const struct list * list )
```

Returns the last node.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

the last node.

**Precondition**

*list* must be not NULL.

**Remarks**

Complexity: O(N)

**4.25.2.15 list\_length()**

```
size_t list_length (
    const struct list * list ) [inline]
```

Returns the size of the list.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

the length of the list.

**Precondition**

*list* must be not NULL.

**Remarks**

Complexity: O(1)

**4.25.2.16 list\_next()**

```
struct list_node* list_next (
    const struct list_node * node )
```

Returns the next node.

**Parameters**

<i>node</i>	a node.
-------------	---------

**Returns**

the next node.

**Precondition**

*node* must be not NULL.

**Remarks**

Complexity: O(1)

**4.25.2.17 list\_print()**

```
void list_print (
    const struct list * list )
```

Print the list

**Parameters**

<i>list</i>	a list
-------------	--------

**4.25.2.18 list\_reverse()**

```
void list_reverse (
    struct list * list ) [inline]
```

Reverses the order of the elements in the list.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Precondition**

`list` must be not NULL.

**Remarks**

Complexity: O(N)

**4.25.2.19 list\_sort()**

```
void list_sort (
    struct list * list,
    int (*) (struct list_node *, struct list_node *) cmp ) [inline]
```

Sort a list using a comparison function.

The contents of the list are sorted in ascending order according to a comparison function which is called with two arguments that point to the node being compared.

The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

If two members compare as equal, their order in the sorted list is preserved.

**Parameters**

<i>list</i>	list to sort.
<i>cmp</i>	comparison function to use.

**Precondition**

`list` must be not NULL.  
`cmp` must be not NULL.

**Remarks**

The sort is stable.  
Complexity: O(N log N)  
Space complexity: O(1)

#### 4.25.2.20 list\_split\_at()

```
void list_split_at (
    struct list * list,
    size_t pos,
    struct list * right ) [inline]
```

Splits a list in two parts at the position *pos*.

After the split:

- *list* contains nodes in [0, *pos*[
- *right* contains nodes in [*pos*,length(*list*)[

Examples:

```
list = [1, 2, 3]
list_split_at(list, 0, right) => ([], [1,2,3])
list_split_at(list, 1, right) => ([1], [2,3])
list_split_at(list, 2, right) => ([1,2], [3])
list_split_at(list, 3, right) => ([1,2,3], [])
list = []
list_split_at(list, 0, right) => ([], [])
```

#### Parameters

<i>list</i>	list to split.
<i>pos</i>	position (0-based) where to split the list.
<i>right</i>	an empty list to receive the part after <i>pos</i>

#### Precondition

*list* must be not NULL.  
*right* must be not NULL.  
*right* must be empty.  
*list* must be different of *right*.

#### Remarks

Complexity: O(N)

#### 4.25.2.21 list\_swap()

```
void list_swap (
    struct list * l1,
    struct list * l2 ) [inline]
```

Swaps two lists.

## Parameters

<i>/1</i>	list 1.
<i>/2</i>	list 2.

## Precondition

*l1* must be not NULL.  
*l2* must be not NULL.  
*l1* must be different of *l2*.

## Remarks

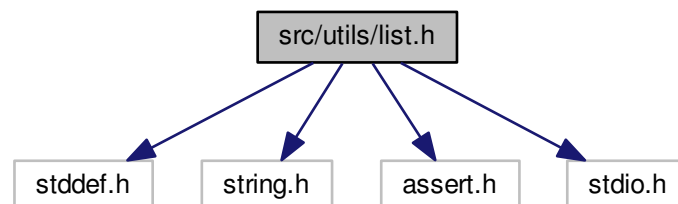
Complexity:  $O(1)$

## 4.26 src/utls/list.h File Reference

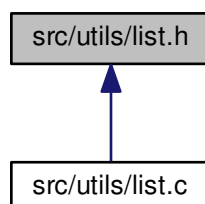
Intrusive list implement.

```
#include <stddef.h>
#include <string.h>
#include <assert.h>
#include <stdio.h>
```

Include dependency graph for list.h:



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



## Data Structures

- struct **list**
- struct **list\_node**

## Macros

- #define **list\_elt**(node, type, fieldname) ((type\*)((char\*)(node) - offsetof(type, fieldname)))
- #define **list\_foreach**( list, curr) for (curr = **list\_first**( list); curr != NULL; curr = **list\_next**(curr))
- #define **list\_foreach\_elt**( list, curr, type, fieldname)
- #define **list\_foreach\_safe**( list, curr, tmp)
- #define **list\_foreach\_elt\_safe**( list, curr, tmp, type, fieldname)

## Functions

- void **list\_init** (struct **list** \* list)
- size\_t **list\_length** (const struct **list** \* list)
- struct **list\_node** \* **list\_first** (const struct **list** \* list)
- struct **list\_node** \* **list\_last** (const struct **list** \* list)
- struct **list\_node** \* **list\_next** (const struct **list\_node** \* node)
- struct **list\_node** \* **list\_advance** (struct **list\_node** \* node, size\_t distance)
- struct **list\_node** \* **list\_at** (const struct **list** \* list, size\_t pos)
- void **list\_reverse** (struct **list** \* list)
- void **list\_swap** (struct **list** \*l1, struct **list** \*l2)
- void **list\_split\_at** (struct **list** \* list, size\_t pos, struct **list** \*right)
- void **list\_concat** (struct **list** \*l1, struct **list** \*l2)
- void **list\_sort** (struct **list** \* list, int(\*cmp)(struct **list\_node** \*, struct **list\_node** \*))
- int **list\_is\_empty** (const struct **list** \* list)
- void **list\_add** (struct **list** \* list, struct **list\_node** \* node)
- void **list\_append** (struct **list** \* list, struct **list\_node** \* node)
- void **list\_insert\_after** (struct **list** \* list, struct **list\_node** \* curr, struct **list\_node** \* node)
- void **list\_insert\_at** (struct **list** \* list, struct **list\_node** \* node, size\_t pos)
- void **list\_del** (struct **list** \* list)
- void **list\_del\_after** (struct **list** \* list, struct **list\_node** \* node)
- void **list\_del\_at** (struct **list** \* list, size\_t pos)
- void **list\_print** (const struct **list** \* list)

### 4.26.1 Detailed Description

Intrusive list implement.

Author

S4MasterRace

Version

1.0



## 4.26.2 Macro Definition Documentation

### 4.26.2.1 list\_elt

```
#define list_elt(  
    node,  
    type,  
    fieldname ) ((type*)((char*)(node) - offsetof(type, fieldname)))
```

Returns a pointer to the structure which contains the node.

#### Parameters

<i>node</i>	a list node (struct list_node*).
<i>type</i>	type of the structure which contains the node.
<i>fieldname</i>	name of the node (field name) in the structure.

#### Precondition

*node* must be not NULL.

#### Remarks

Complexity: O(1)

### 4.26.2.2 list\_foreach

```
#define list_foreach(  
    list,  
    curr ) for (curr = list_first( list); curr != NULL; curr = list_next(curr))
```

Iterates over list (nodes).

#### Parameters

<i>list</i>	a list (struct list*).
<i>curr</i>	a struct list_node* used to hold the current element.

#### Precondition

*list* must be not NULL.  
*curr* must be not NULL.

**Remarks**

Complexity: O(N)

**4.26.2.3 list\_foreach\_elt**

```
#define list_foreach_elt(
    list,
    curr,
    type,
    fieldname )
```

**Value:**

```
for (curr = list_elt(list_first(list), type, fieldname);    \
     curr != NULL;                                         \
     curr = curr->fieldname.next == NULL ? NULL :         \
     list_elt(list_next(&(curr->fieldname)), type, fieldname))
```

Iterates over list (elements)

**Parameters**

<i>list</i>	a list (struct list*).
<i>curr</i>	pointer (type*) used to hold the current element.
<i>type</i>	type of the structure which contains the node.
<i>fieldname</i>	name of the node (field name) in the structure.

**Precondition**

*list* must be not NULL.  
*list* must be not empty.  
*curr* must be not NULL.

**Remarks**

Complexity: O(N)

**4.26.2.4 list\_foreach\_elt\_safe**

```
#define list_foreach_elt_safe(
    list,
    curr,
    tmp,
    type,
    fieldname )
```

**Value:**

```

for (curr = list_elt(list_first(list), type, fieldname), \
     tmp = list_next(&(curr->fieldname)); \
     curr != NULL; \
     curr = tmp == NULL ? NULL : list_elt(tmp, type, fieldname), \
     tmp = tmp == NULL ? NULL : list_next(tmp))

```

Iterates over list (elements), allows deletion of the current element.

#### Parameters

<i>list</i>	a list (struct list*).
<i>curr</i>	pointer (type*) used to hold the current element.
<i>tmp</i>	a struct list_node* used as temporary storage.
<i>type</i>	type of the structure which contains the node.
<i>fieldname</i>	name of the node (field name) in the structure.

#### Precondition

list must be not NULL.  
 list must be not empty.  
 curr must be not NULL.

#### Remarks

Complexity: O(N)

#### 4.26.2.5 list\_foreach\_safe

```

#define list_foreach_safe(
    list,
    curr,
    tmp )

```

#### Value:

```

for (curr = list_first(list), tmp = list_next(curr); \
     curr != NULL; \
     curr = tmp, tmp = tmp == NULL ? NULL : list_next(tmp))

```

Iterates over list (nodes), allows deletion of the current node.

#### Parameters

<i>list</i>	a list (struct list*).
<i>curr</i>	a struct list_node* used to hold the current element.
<i>tmp</i>	a struct list_node* used as temporary storage.

**Precondition**

`list` must be not NULL.  
`curr` must be not NULL.  
`tmp` must be not NULL.

**Remarks**

Complexity: O(N)

**4.26.3 Function Documentation****4.26.3.1 `list_add()`**

```
void list_add (  
    struct list * list,  
    struct list_node * node ) [inline]
```

Adds `node` in the front of `list`

**Parameters**

<i>list</i>	a list.
<i>node</i>	the new node.

**Precondition**

`list` must be not NULL.  
`node` must be not NULL.

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(1)

**4.26.3.2 `list_advance()`**

```
struct list_node* list_advance (  
    struct list_node * node,  
    size_t distance )
```

Returns the `n`th-node after the current one.

**Parameters**

<i>node</i>	a node.
<i>distance</i>	distance to move on.

**Returns**

the nth-node after *node*.

**Precondition**

*node* must be not NULL.

**Remarks**

Complexity: O(n)

**4.26.3.3 list\_append()**

```
void list_append (  
    struct list * list,  
    struct list_node * node ) [inline]
```

Adds *node* at the end of *list*.

**Parameters**

<i>list</i>	a list.
<i>node</i>	the new node.

**Precondition**

*list* must be not NULL.  
*node* must be not NULL.

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(n)

#### 4.26.3.4 list\_at()

```
struct list_node* list_at (  
    const struct list * list,  
    size_t pos )
```

Returns node at the position `pos`.

**Parameters**

<i>list</i>	a list.
<i>pos</i>	position (0-based) of the node.

**Returns**

the node at the position `pos`.

**Precondition**

`list` must be not NULL.  
`list` must be not empty.  
`pos` must be in `[0; list_length(list)[`.

**Remarks**

Complexity:  $O(N)$

**4.26.3.5 list\_concat()**

```
void list_concat (
    struct list * l1,
    struct list * l2 ) [inline]
```

Concatenates two lists.

**Parameters**

<i>l1</i>	list 1.
<i>l2</i>	list 2.

**Precondition**

`l1` must be not NULL.  
`l2` must be not NULL.  
`l1` must be different of `l2`.

**Postcondition**

`l2` is reset to an empty list.

**Remarks**

Complexity:  $O(N)$

#### 4.26.3.6 list\_del()

```
void list_del (
    struct list * list ) [inline]
```

Deletes the first node.

##### Parameters

<i>list</i>	a list.
-------------	---------

##### Precondition

*list* must be not NULL.  
*list* must be not empty.

##### Postcondition

List size decreases by 1.

##### Remarks

Complexity: O(1)

#### 4.26.3.7 list\_del\_after()

```
void list_del_after (
    struct list * list,
    struct list_node * node ) [inline]
```

Deletes the node at after the node *curr*.

##### Parameters

<i>list</i>	a list.
<i>node</i>	a node of <i>list</i> .

##### Precondition

*list* must be not NULL.  
*node* must be not NULL.  
*list* must be not empty.  
*node* must a node of *list*.

##### Postcondition

List size decreases by 1.



**Remarks**

Complexity:  $O(1)$

**4.26.3.8 list\_del\_at()**

```
void list_del_at (
    struct list * list,
    size_t pos ) [inline]
```

Deletes the node at the position *pos*.

**Parameters**

<i>list</i>	a list.
<i>pos</i>	index (0-based) of the node to delete.

**Precondition**

*list* must be not NULL.  
*list* must be not empty.  
*pos* must be in  $[0; \text{list\_length}(\text{list})[$ .

**Postcondition**

List size decreases by 1.

**Remarks**

Complexity:  $O(n)$

**4.26.3.9 list\_first()**

```
struct list_node* list_first (
    const struct list * list )
```

Returns the first node.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

the first node.

**Precondition**

`list` must be not NULL.  
`list` must be not empty.

**Remarks**

Complexity: O(1)

**4.26.3.10 list\_init()**

```
void list_init (
    struct list * list ) [inline]
```

Initializes the list.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Precondition**

`list` must be not NULL.

**Postcondition**

`list` is empty.  
`list` has a size of 0.

**Remarks**

Complexity: O(1)

**4.26.3.11 list\_insert\_after()**

```
void list_insert_after (
    struct list * list,
    struct list_node * curr,
    struct list_node * node ) [inline]
```

Inserts `node` at after the node `curr`.

**Parameters**

<i>list</i>	a list.
<i>curr</i>	a node of <code>list</code> .
<i>node</i>	new node.

**Precondition**

`list` must be not NULL.  
`curr` must be not NULL.  
`curr` must a node of `list`.  
`node` must be not NULL.

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(1)

**4.26.3.12 list\_insert\_at()**

```
void list_insert_at (
    struct list * list,
    struct list_node * node,
    size_t pos ) [inline]
```

Inserts `node` at the position `pos` in `list`.

**Parameters**

<i>list</i>	a list.
<i>node</i>	new node.
<i>pos</i>	position (0-based) where to insert the new node.

**Precondition**

`list` must be not NULL.  
`node` must be not NULL.  
`pos` must be in [0; list\_length(list)].

**Postcondition**

List size increases by 1.

**Remarks**

Complexity: O(n)

**4.26.3.13 list\_is\_empty()**

```
int list_is_empty (
    const struct list * list ) [inline]
```

Tests if a list is empty.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

1 if the list is empty, otherwise 0.

**Precondition**

`list` must be not NULL.

**Remarks**

Complexity: O(1)

**4.26.3.14 list\_last()**

```
struct list_node* list_last (
    const struct list * list )
```

Returns the last node.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

the last node.

**Precondition**

`list` must be not NULL.

**Remarks**

Complexity: O(N)

**4.26.3.15 list\_length()**

```
size_t list_length (
    const struct list * list ) [inline]
```

Returns the size of the list.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Returns**

the length of the list.

**Precondition**

`list` must be not NULL.

**Remarks**

Complexity: O(1)

**4.26.3.16 list\_next()**

```
struct list_node* list_next (
    const struct list_node * node )
```

Returns the next node.

**Parameters**

<i>node</i>	a node.
-------------	---------

**Returns**

the next node.

**Precondition**

`node` must be not NULL.

**Remarks**

Complexity: O(1)

**4.26.3.17 list\_print()**

```
void list_print (
    const struct list * list )
```

Print the list

**Parameters**

<i>list</i>	a list
-------------	--------

**4.26.3.18 list\_reverse()**

```
void list_reverse (
    struct list * list ) [inline]
```

Reverses the order of the elements in the list.

**Parameters**

<i>list</i>	a list.
-------------	---------

**Precondition**

*list* must be not NULL.

**Remarks**

Complexity: O(N)

**4.26.3.19 list\_sort()**

```
void list_sort (
    struct list * list,
    int (*) (struct list_node *, struct list_node *) cmp ) [inline]
```

Sort a list using a comparison function.

The contents of the list are sorted in ascending order according to a comparison function which is called with two arguments that point to the node being compared.

The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

If two members compare as equal, their order in the sorted list is preserved.

**Parameters**

<i>list</i>	list to sort.
<i>cmp</i>	comparison function to use.

**Precondition**

`list` must be not NULL.  
`cmp` must be not NULL.

**Remarks**

The sort is stable.  
 Complexity:  $O(N \log N)$   
 Space complexity:  $O(1)$

**4.26.3.20 list\_split\_at()**

```
void list_split_at (
    struct list * list,
    size_t pos,
    struct list * right ) [inline]
```

Splits a list in two parts at the position `pos`.

After the split:

- `list` contains nodes in `[0, pos[`
- `right` contains nodes in `[pos,length(list)[`

**Examples:**

```
list = [1, 2, 3]
list_split_at(list, 0, right) => ([], [1,2,3])
list_split_at(list, 1, right) => ([1], [2,3])
list_split_at(list, 2, right) => ([1,2], [3])
list_split_at(list, 3, right) => ([1,2,3], [])
list = []
list_split_at(list, 0, right) => ([], [])
```

**Parameters**

<i>list</i>	list to split.
<i>pos</i>	position (0-based) where to split the list.
<i>right</i>	an empty list to receive the part after <code>pos</code>

**Precondition**

`list` must be not NULL.  
`right` must be not NULL.  
`right` must be empty.  
`list` must be different of `right`.

**Remarks**

Complexity:  $O(N)$

**4.26.3.21 list\_swap()**

```
void list_swap (
    struct list * l1,
    struct list * l2 ) [inline]
```

Swaps two lists.

**Parameters**

<i>l1</i>	list 1.
<i>l2</i>	list 2.

**Precondition**

*l1* must be not NULL.  
*l2* must be not NULL.  
*l1* must be different of *l2*.

**Remarks**

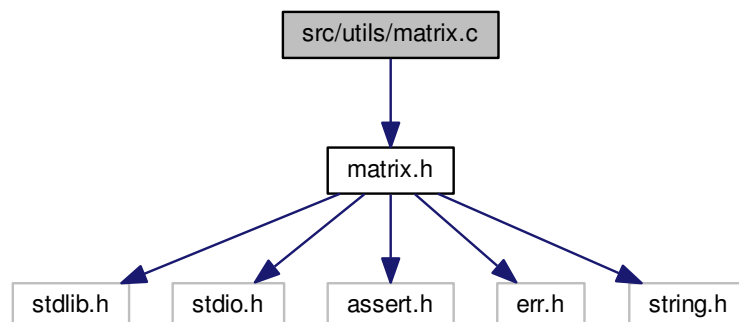
Complexity:  $O(1)$

**4.27 src/utils/matrix.c File Reference**

Matrix implement.

```
#include "matrix.h"
```

Include dependency graph for matrix.c:





## Functions

- struct **matrix** \* **matrix\_create** (size\_t rows, size\_t cols)
- struct **matrix** \* **matrix\_create\_from\_array** (size\_t rows, size\_t cols, const double values[ ])
- void **matrix\_free** (struct **matrix** \*mx)
- size\_t **matrix\_rows** (const struct **matrix** \*mx)
- size\_t **matrix\_cols** (const struct **matrix** \*mx)
- double **matrix\_at** (const struct **matrix** \*mx, size\_t rows, size\_t cols)
- void **matrix\_set** (struct **matrix** \*mx, size\_t rows, size\_t cols, double value)
- struct **matrix** \* **matrix\_copy** (const struct **matrix** \*mx)
- void **matrix\_transpose** (const struct **matrix** \*mx, struct **matrix** \*tmx)
- void **matrix\_sum** (const struct **matrix** \*mx1, const struct **matrix** \*mx2, struct **matrix** \*sum)
- void **matrix\_hadamard\_product** (const struct **matrix** \*mx1, const struct **matrix** \*mx2, struct **matrix** \*prod)
- void **matrix\_product** (const struct **matrix** \*mx1, const struct **matrix** \*mx2, struct **matrix** \*prod)
- void **matrix\_scale** (const struct **matrix** \*mx, double scale, struct **matrix** \*smx)
- double **matrix\_dot\_product** (const struct **matrix** \*v1, const struct **matrix** \*v2)
- void **matrix\_identity** (struct **matrix** \*mx)
- int **matrix\_is\_square** (const struct **matrix** \*mx)
- int **matrix\_is\_diagonal** (const struct **matrix** \*mx)
- int **matrix\_is\_upper\_triangular** (const struct **matrix** \*mx)
- void **matrix\_diagonal** (const struct **matrix** \*v, struct **matrix** \*mx)
- void **matrix\_print** (const struct **matrix** \*mx)

### 4.27.1 Detailed Description

Matrix implement.

Author

S4MasterRace

Version

1.0

### 4.27.2 Function Documentation

#### 4.27.2.1 matrix\_at()

```
double matrix_at (
    const struct matrix * mx,
    size_t rows,
    size_t cols ) [inline]
```

Get value at rows rows and cols columns of mx

**Parameters**

<i>mx</i>	a matrix
<i>rows</i>	rows
<i>cols</i>	columns

**Returns**

the value at `rows` rows and `cols` columns of `mx`

**Precondition**

`mx` must be not NULL  
`rows` must be between `[0, matrix_rows(mx) [`  
`cols` must be between `[0, matrix_cols(mx) [`

**Remarks**

Complexity: O(1)

**4.27.2.2 matrix\_cols()**

```
size_t matrix_cols (
    const struct matrix * mx ) [inline]
```

Get the number of columns of `mx`

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

the number of columns `mx`

**Precondition**

`mx` must be not NULL

**Remarks**

Complexity: O(1)

**4.27.2.3 matrix\_copy()**

```
struct matrix* matrix_copy (
    const struct matrix * mx )
```

Copy the matrix `mx`

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

the copy of *mx*

**Precondition**

*mx* must be not NULL

**Remarks**

Complexity: O(1)

**4.27.2.4 matrix\_create()**

```
struct matrix* matrix_create (  
    size_t rows,  
    size_t cols )
```

Create a matrix of size *rows* rows and *cols* columns

**Parameters**

<i>rows</i>	number of rows
<i>cols</i>	number of columns

**Returns**

the initialized matrix of size *rows* rows and *cols* columns

**Precondition**

*rows* must be greater than zero  
*cols* must be greater than zero

**Remarks**

Complexity: O(1)

#### 4.27.2.5 matrix\_create\_from\_array()

```
struct matrix* matrix_create_from_array (
    size_t rows,
    size_t cols,
    const double values[] )
```

Create a matrix of size `rows` rows and `cols` columns from `values`

**Parameters**

<i>rows</i>	number of rows
<i>cols</i>	number of columns
<i>values</i>	an array

**Returns**

the initialized matrix of size `rows` rows and `cols` columns from `values`

**Precondition**

`rows` must be greater than zero  
`cols` must be greater than zero  
`values` must be not NULL

**Remarks**

Complexity: O(N)

**4.27.2.6 matrix\_diagonal()**

```
void matrix_diagonal (
    const struct matrix * v,
    struct matrix * mx )
```

Take a vector `v` and put it to the diagonal of `mx`

**Parameters**

<i>v</i>	a vector
<i>mx</i>	a matrix

**Precondition**

`v` must be not NULL  
`mx` must be not NULL  
`matrix_cols(v)` must be equal to one  
`matrix_rows(v)` must be equal to `matrix_rows(mx)`  
`matrix_rows(mx)` must be equal to `matrix_cols(mx)`

**Postcondition**

The diagonal of `mx` is `v`

#### 4.27.2.7 matrix\_dot\_product()

```
double matrix_dot_product (
    const struct matrix * v1,
    const struct matrix * v2 )
```

Do the dot product of vector `v1` with vector `v2`

##### Parameters

<code>v1</code>	a vector
<code>v2</code>	a vector

##### Returns

the dot product of vector `v1` with vector `v2`

##### Precondition

`v1` must be not NULL  
`v2` must be not NULL  
`matrix_cols(v1)` and `matrix_cols(v2)` must be equal to one  
`matrix_rows(v1)` must be equal to `matrix_rows(v2)`

##### Remarks

Complexity:  $O(N)$

#### 4.27.2.8 matrix\_free()

```
void matrix_free (
    struct matrix * mx ) [inline]
```

Free the matrix `mx`

##### Parameters

<code>mx</code>	a matrix
-----------------	----------

##### Precondition

`mx` must be not NULL

##### Postcondition

`mx` is freed

**Remarks**

Complexity:  $O(1)$

**4.27.2.9 matrix\_hadamard\_product()**

```
void matrix_hadamard_product (
    const struct matrix * mx1,
    const struct matrix * mx2,
    struct matrix * prod )
```

Do the hadamard product of matrix *mx1* with *mx2*

**Parameters**

<i>mx1</i>	a matrix
<i>mx2</i>	a matrix
<i>prod</i>	a matrix

**Precondition**

*mx1* must be not NULL  
*mx2* must be not NULL  
*prod* must be not NULL  
*matrix\_rows*(*mx1*) must be equal to *matrix\_rows*(*mx2*)  
*matrix\_cols*(*mx1*) must be equal to *matrix\_cols*(*mx2*)  
*matrix\_rows*(*prod*) must be equal to *matrix\_rows*(*mx1*)  
*matrix\_cols*(*prod*) must be equal to *matrix\_cols*(*mx1*)

**Postcondition**

*prod* is the hadamard product of matrix *mx1* with *mx2*

**Remarks**

Complexity:  $O(N)$

**4.27.2.10 matrix\_identity()**

```
void matrix_identity (
    struct matrix * mx )
```

Set the matrix *mx* to an identity matrix

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Precondition**

*mx* must be not NULL  
`matrix_rows(mx)` must be equal to `matrix_cols(mx)`

**Postcondition**

*mx* is an identity matrix

**Remarks**

Complexity: O(N)

**4.27.2.11 `matrix_is_diagonal()`**

```
int matrix_is_diagonal (  
    const struct matrix * mx ) [inline]
```

Check if the matrix *mx* is diagonaled

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

1 if the matrix is diagonaled, 0 otherwise

**Precondition**

*mx* must be not NULL

**Remarks**

Complexity: O(N)

**4.27.2.12 `matrix_is_square()`**

```
int matrix_is_square (  
    const struct matrix * mx ) [inline]
```

Check if the matrix *mx* is squared



**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

1 if the matrix is squared, 0 otherwise

**Precondition**

*mx* must be not NULL

**Remarks**

Complexity: O(1)

**4.27.2.13 matrix\_is\_upper\_triangular()**

```
int matrix_is_upper_triangular (
    const struct matrix * mx ) [inline]
```

Check if the matrix *mx* is upper triangular

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

1 if the matrix is upper triangular, 0 otherwise

**Precondition**

*mx* must be not NULL

**Remarks**

Complexity: O(N)

**4.27.2.14 matrix\_print()**

```
void matrix_print (
    const struct matrix * mx )
```

Print the matrix *mx*

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Precondition**

*mx* must be not NULL

**Postcondition**

Print the matrix *mx*

**4.27.2.15 matrix\_product()**

```
void matrix_product (
    const struct matrix * mx1,
    const struct matrix * mx2,
    struct matrix * prod )
```

Multiply the matrix *mx1* with *mx2*

**Parameters**

<i>mx1</i>	a matrix
<i>mx2</i>	a matrix
<i>prod</i>	a matrix

**Precondition**

*mx1* must be not NULL  
*mx2* must be not NULL  
*prod* must be not NULL  
*prod* must be not equal to *mx1*  
*prod* must be not equal to *mx2*  
*matrix\_cols*(*mx1*) must be equal to *matrix\_rows*(*mx2*)  
*matrix\_rows*(*prod*) must be equal to *matrix\_rows*(*mx1*)  
*matrix\_cols*(*prod*) must be equal to *matrix\_cols*(*mx2*)

**Postcondition**

*prod* is the product of *mx1* with *mx2*

**Remarks**

Complexity: O(nmp)

## 4.27.2.16 matrix\_rows()

```
size_t matrix_rows (
    const struct matrix * mx ) [inline]
```

Get the number of rows *mx*

## Parameters

<i>mx</i>	a matrix
-----------	----------

## Returns

the number of rows of *mx*

## Precondition

*mx* must be not NULL

## Remarks

Complexity: O(1)

## 4.27.2.17 matrix\_scale()

```
void matrix_scale (
    const struct matrix * mx,
    double scale,
    struct matrix * smx )
```

Scale the matrix *mx* with *scale*

## Parameters

<i>mx</i>	a matrix
<i>scale</i>	the scale factor
<i>smx</i>	a matrix

## Precondition

*mx* must be not NULL

*smx* must be not NULL

matrix\_rows(*smx*) must be equal to matrix\_rows(*mx*)

matrix\_cols(*smx*) must be equal to matrix\_cols(*mx*)

## Postcondition

*smx* is the *scale* scaled matrix of *mx*

**Remarks**

Complexity: O(N)

**4.27.2.18 matrix\_set()**

```
void matrix_set (
    struct matrix * mx,
    size_t rows,
    size_t cols,
    double value ) [inline]
```

Set the value at `rows` rows and `cols` columns with value of `mx`

**Parameters**

<i>mx</i>	a matrix
<i>rows</i>	rows
<i>cols</i>	columns
<i>value</i>	a value

**Precondition**

`mx` must be not NULL  
`rows` must be between `[0, matrix_rows(mx) [`  
`cols` must be between `[0, matrix_cols(mx) [`

**Postcondition**

the value at `rows` rows and `cols` columns is `value`

**Remarks**

Complexity: O(1)

**4.27.2.19 matrix\_sum()**

```
void matrix_sum (
    const struct matrix * mx1,
    const struct matrix * mx2,
    struct matrix * sum )
```

Sum the matrix `mx1` with `mx2`

**Parameters**

<i>mx1</i>	a matrix
<i>mx2</i>	a matrix
<i>sum</i>	a matrix

**Precondition**

`mx1` must be not NULL  
`mx2` must be not NULL  
`sum` must be not NULL  
`matrix_rows(mx1)` must be equal to `matrix_rows(mx2)`  
`matrix_cols(mx1)` must be equal to `matrix_cols(mx2)`  
`matrix_rows(sum)` must be equal to `matrix_rows(mx1)`  
`matrix_cols(sum)` must be equal to `matrix_cols(mx1)`

**Postcondition**

`sum` is the sum of matrix `mx1` with `mx2`

**Remarks**

Complexity: O(N)

**4.27.2.20 matrix\_transpose()**

```
void matrix_transpose (
    const struct matrix * mx,
    struct matrix * tmx )
```

Transpose the matrix `mx`

**Parameters**

<i>mx</i>	a matrix
<i>tmx</i>	a matrix

**Precondition**

`mx` must be not NULL  
`tmx` must be not NULL  
`tmx` must be not equal to `mx`  
`matrix_rows(tmx)` must be equal to `matrix_cols(mx)`  
`matrix_cols(tmx)` must be equal to `matrix_rows(mx)`

**Postcondition**

`tmx` is the transposed matrix of `mx`

**Remarks**

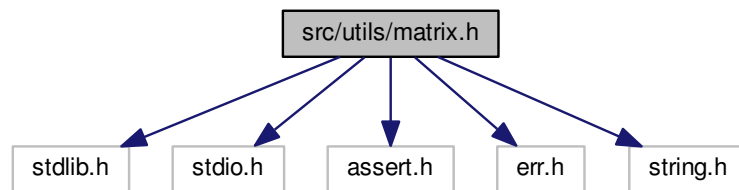
Complexity: O(N)

## 4.28 src/utils/matrix.h File Reference

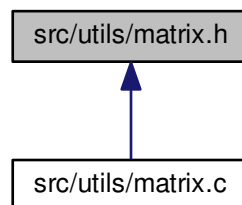
Matrix implement.

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <err.h>
#include <string.h>
```

Include dependency graph for matrix.h:



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



### Data Structures

- struct **matrix**

### Functions

- struct **matrix** \* **matrix\_create** (size\_t rows, size\_t cols)
- struct **matrix** \* **matrix\_create\_from\_array** (size\_t rows, size\_t cols, const double values[])
- void **matrix\_free** (struct **matrix** \*mx)
- size\_t **matrix\_rows** (const struct **matrix** \*mx)
- size\_t **matrix\_cols** (const struct **matrix** \*mx)
- double **matrix\_at** (const struct **matrix** \*mx, size\_t rows, size\_t cols)

- void **matrix\_set** (struct **matrix** \*mx, size\_t rows, size\_t cols, double value)
- struct **matrix** \* **matrix\_copy** (const struct **matrix** \*mx)
- void **matrix\_transpose** (const struct **matrix** \*mx, struct **matrix** \*tmx)
- void **matrix\_sum** (const struct **matrix** \*mx1, const struct **matrix** \*mx2, struct **matrix** \*sum)
- void **matrix\_hadamard\_product** (const struct **matrix** \*mx1, const struct **matrix** \*mx2, struct **matrix** \*prod)
- void **matrix\_product** (const struct **matrix** \*mx1, const struct **matrix** \*mx2, struct **matrix** \*prod)
- void **matrix\_scale** (const struct **matrix** \*mx, double scale, struct **matrix** \*smx)
- double **matrix\_dot\_product** (const struct **matrix** \*v1, const struct **matrix** \*v2)
- void **matrix\_identity** (struct **matrix** \*mx)
- int **matrix\_is\_square** (const struct **matrix** \*mx)
- int **matrix\_is\_diagonal** (const struct **matrix** \*mx)
- int **matrix\_is\_upper\_triangular** (const struct **matrix** \*mx)
- void **matrix\_diagonal** (const struct **matrix** \*v, struct **matrix** \*mx)
- void **matrix\_print** (const struct **matrix** \*mx)

### 4.28.1 Detailed Description

Matrix implement.

Author

S4MasterRace

Version

1.0

### 4.28.2 Function Documentation

#### 4.28.2.1 matrix\_at()

```
double matrix_at (  
    const struct matrix * mx,  
    size_t rows,  
    size_t cols ) [inline]
```

Get value at `rows` rows and `cols` columns of `mx`

Parameters

<i>mx</i>	a matrix
<i>rows</i>	rows
<i>cols</i>	columns

Returns

the value at `rows` rows and `cols` columns of `mx`

**Precondition**

`mx` must be not NULL  
rows must be between `[0, matrix_rows(mx) [`  
cols must be between `[0, matrix_cols(mx) [`

**Remarks**

Complexity: O(1)

**4.28.2.2 matrix\_cols()**

```
size_t matrix_cols (  
    const struct matrix * mx ) [inline]
```

Get the number of columns of `mx`

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

the number of columns `mx`

**Precondition**

`mx` must be not NULL

**Remarks**

Complexity: O(1)

**4.28.2.3 matrix\_copy()**

```
struct matrix* matrix_copy (  
    const struct matrix * mx )
```

Copy the matrix `mx`

**Parameters**

<i>mx</i>	a matrix
-----------	----------



**Returns**

the copy of `mx`

**Precondition**

`mx` must be not NULL

**Remarks**

Complexity: O(1)

**4.28.2.4 matrix\_create()**

```
struct matrix* matrix_create (  
    size_t rows,  
    size_t cols )
```

Create a matrix of size `rows` rows and `cols` columns

**Parameters**

<i>rows</i>	number of rows
<i>cols</i>	number of columns

**Returns**

the initialized matrix of size `rows` rows and `cols` columns

**Precondition**

`rows` must be greater than zero  
`cols` must be greater than zero

**Remarks**

Complexity: O(1)

**4.28.2.5 matrix\_create\_from\_array()**

```
struct matrix* matrix_create_from_array (  
    size_t rows,  
    size_t cols,  
    const double values[] )
```

Create a matrix of size `rows` rows and `cols` columns from `values`

**Parameters**

<i>rows</i>	number of rows
<i>cols</i>	number of columns
<i>values</i>	an array

**Returns**

the initialized matrix of size `rows` rows and `cols` columns from `values`

**Precondition**

`rows` must be greater than zero  
`cols` must be greater than zero  
`values` must be not NULL

**Remarks**

Complexity: O(N)

**4.28.2.6 matrix\_diagonal()**

```
void matrix_diagonal (
    const struct matrix * v,
    struct matrix * mx )
```

Take a vector `v` and put it to the diagonal of `mx`

**Parameters**

<i>v</i>	a vector
<i>mx</i>	a matrix

**Precondition**

`v` must be not NULL  
`mx` must be not NULL  
`matrix_cols(v)` must be equal to one  
`matrix_rows(v)` must be equal to `matrix_rows(mx)`  
`matrix_rows(mx)` must be equal to `matrix_cols(mx)`

**Postcondition**

The diagonal of `mx` is `v`

## 4.28.2.7 matrix\_dot\_product()

```
double matrix_dot_product (
    const struct matrix * v1,
    const struct matrix * v2 )
```

Do the dot product of vector `v1` with vector `v2`

## Parameters

<code>v1</code>	a vector
<code>v2</code>	a vector

## Returns

the dot product of vector `v1` with vector `v2`

## Precondition

`v1` must be not NULL  
`v2` must be not NULL  
`matrix_cols(v1)` and `matrix_cols(v2)` must be equal to one  
`matrix_rows(v1)` must be equal to `matrix_rows(v2)`

## Remarks

Complexity:  $O(N)$

## 4.28.2.8 matrix\_free()

```
void matrix_free (
    struct matrix * mx ) [inline]
```

Free the matrix `mx`

## Parameters

<code>mx</code>	a matrix
-----------------	----------

## Precondition

`mx` must be not NULL

## Postcondition

`mx` is freed

**Remarks**

Complexity: O(1)

**4.28.2.9 matrix\_hadamard\_product()**

```
void matrix_hadamard_product (
    const struct matrix * mx1,
    const struct matrix * mx2,
    struct matrix * prod )
```

Do the hadamard product of matrix *mx1* with *mx2*

**Parameters**

<i>mx1</i>	a matrix
<i>mx2</i>	a matrix
<i>prod</i>	a matrix

**Precondition**

*mx1* must be not NULL  
*mx2* must be not NULL  
*prod* must be not NULL  
*matrix\_rows*(*mx1*) must be equal to *matrix\_rows*(*mx2*)  
*matrix\_cols*(*mx1*) must be equal to *matrix\_cols*(*mx2*)  
*matrix\_rows*(*prod*) must be equal to *matrix\_rows*(*mx1*)  
*matrix\_cols*(*prod*) must be equal to *matrix\_cols*(*mx1*)

**Postcondition**

*prod* is the hadamard product of matrix *mx1* with *mx2*

**Remarks**

Complexity: O(N)

**4.28.2.10 matrix\_identity()**

```
void matrix_identity (
    struct matrix * mx )
```

Set the matrix *mx* to an identity matrix

#### Parameters

<i>mx</i>	a matrix
-----------	----------

#### Precondition

*mx* must be not NULL  
`matrix_rows(mx)` must be equal to `matrix_cols(mx)`

#### Postcondition

*mx* is an identity matrix

#### Remarks

Complexity: O(N)

#### 4.28.2.11 `matrix_is_diagonal()`

```
int matrix_is_diagonal (  
    const struct matrix * mx ) [inline]
```

Check if the matrix *mx* is diagonaled

#### Parameters

<i>mx</i>	a matrix
-----------	----------

#### Returns

1 if the matrix is diagonaled, 0 otherwise

#### Precondition

*mx* must be not NULL

#### Remarks

Complexity: O(N)

#### 4.28.2.12 `matrix_is_square()`

```
int matrix_is_square (  
    const struct matrix * mx ) [inline]
```

Check if the matrix *mx* is squared

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

1 if the matrix is squared, 0 otherwise

**Precondition**

*mx* must be not NULL

**Remarks**

Complexity: O(1)

**4.28.2.13 matrix\_is\_upper\_triangular()**

```
int matrix_is_upper_triangular (
    const struct matrix * mx ) [inline]
```

Check if the matrix *mx* is upper triangular

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Returns**

1 if the matrix is upper triangular, 0 otherwise

**Precondition**

*mx* must be not NULL

**Remarks**

Complexity: O(N)

**4.28.2.14 matrix\_print()**

```
void matrix_print (
    const struct matrix * mx )
```

Print the matrix *mx*

**Parameters**

<i>mx</i>	a matrix
-----------	----------

**Precondition**

*mx* must be not NULL

**Postcondition**

Print the matrix *mx*

**4.28.2.15 matrix\_product()**

```
void matrix_product (
    const struct matrix * mx1,
    const struct matrix * mx2,
    struct matrix * prod )
```

Multiply the matrix *mx1* with *mx2*

**Parameters**

<i>mx1</i>	a matrix
<i>mx2</i>	a matrix
<i>prod</i>	a matrix

**Precondition**

*mx1* must be not NULL  
*mx2* must be not NULL  
*prod* must be not NULL  
*prod* must be not equal to *mx1*  
*prod* must be not equal to *mx2*  
*matrix\_cols*(*mx1*) must be equal to *matrix\_rows*(*mx2*)  
*matrix\_rows*(*prod*) must be equal to *matrix\_rows*(*mx1*)  
*matrix\_cols*(*prod*) must be equal to *matrix\_cols*(*mx2*)

**Postcondition**

*prod* is the product of *mx1* with *mx2*

**Remarks**

Complexity: O(nmp)

#### 4.28.2.16 `matrix_rows()`

```
size_t matrix_rows (
    const struct matrix * mx ) [inline]
```

Get the number of rows `mx`

##### Parameters

<i>mx</i>	a matrix
-----------	----------

##### Returns

the number of rows of `mx`

##### Precondition

`mx` must be not NULL

##### Remarks

Complexity: O(1)

#### 4.28.2.17 `matrix_scale()`

```
void matrix_scale (
    const struct matrix * mx,
    double scale,
    struct matrix * smx )
```

Scale the matrix `mx` with `scale`

##### Parameters

<i>mx</i>	a matrix
<i>scale</i>	the scale factor
<i>smx</i>	a matrix

##### Precondition

`mx` must be not NULL

`smx` must be not NULL

`matrix_rows (smx)` must be equal to `matrix_rows (mx)`

`matrix_cols (smx)` must be equal to `matrix_cols (mx)`

##### Postcondition

`smx` is the `scale` scaled matrix of `mx`



## Remarks

Complexity: O(N)

## 4.28.2.18 matrix\_set()

```
void matrix_set (
    struct matrix * mx,
    size_t rows,
    size_t cols,
    double value ) [inline]
```

Set the value at rows *rows* and cols columns with value of *mx*

## Parameters

<i>mx</i>	a matrix
<i>rows</i>	rows
<i>cols</i>	columns
<i>value</i>	a value

## Precondition

*mx* must be not NULL  
*rows* must be between [0, matrix\_rows(*mx*) [  
*cols* must be between [0, matrix\_cols(*mx*) [

## Postcondition

the value at rows *rows* and cols columns is *value*

## Remarks

Complexity: O(1)

## 4.28.2.19 matrix\_sum()

```
void matrix_sum (
    const struct matrix * mx1,
    const struct matrix * mx2,
    struct matrix * sum )
```

Sum the matrix *mx1* with *mx2*

## Parameters

<i>mx1</i>	a matrix
<i>mx2</i>	a matrix
<i>sum</i>	a matrix

**Precondition**

`mx1` must be not NULL  
`mx2` must be not NULL  
`sum` must be not NULL  
`matrix_rows(mx1)` must be equal to `matrix_rows(mx2)`  
`matrix_cols(mx1)` must be equal to `matrix_cols(mx2)`  
`matrix_rows(sum)` must be equal to `matrix_rows(mx1)`  
`matrix_cols(sum)` must be equal to `matrix_cols(mx1)`

**Postcondition**

`sum` is the sum of matrix `mx1` with `mx2`

**Remarks**

Complexity:  $O(N)$

**4.28.2.20 `matrix_transpose()`**

```
void matrix_transpose (
    const struct matrix * mx,
    struct matrix * tmx )
```

Transpose the matrix `mx`

**Parameters**

<i>mx</i>	a matrix
<i>tmx</i>	a matrix

**Precondition**

`mx` must be not NULL  
`tmx` must be not NULL  
`tmx` must be not equal to `mx`  
`matrix_rows(tmx)` must be equal to `matrix_cols(mx)`  
`matrix_cols(tmx)` must be equal to `matrix_rows(mx)`

**Postcondition**

`tmx` is the transposed matrix of `mx`

**Remarks**

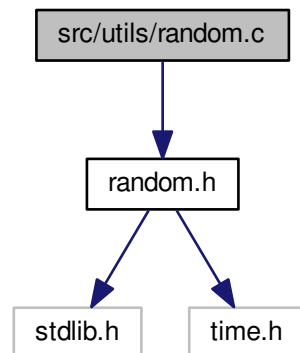
Complexity:  $O(N)$

## 4.29 src/utls/random.c File Reference

No description.

```
#include "random.h"
```

Include dependency graph for random.c:



### Functions

- void **random\_init** ()
- size\_t **random\_size\_t** (size\_t min, size\_t max)
- int **random\_int** (int min, int max)

#### 4.29.1 Detailed Description

No description.

##### Author

S4MasterRace

##### Version

1.0

#### 4.29.2 Function Documentation

#### 4.29.2.1 random\_init()

```
void random_init ( ) [inline]
```

#### 4.29.2.2 random\_int()

```
int random_int (
    int min,
    int max ) [inline]
```

#### 4.29.2.3 random\_size\_t()

```
size_t random_size_t (
    size_t min,
    size_t max ) [inline]
```

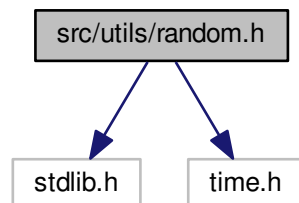
### 4.30 src/utils/random.h File Reference

No description.

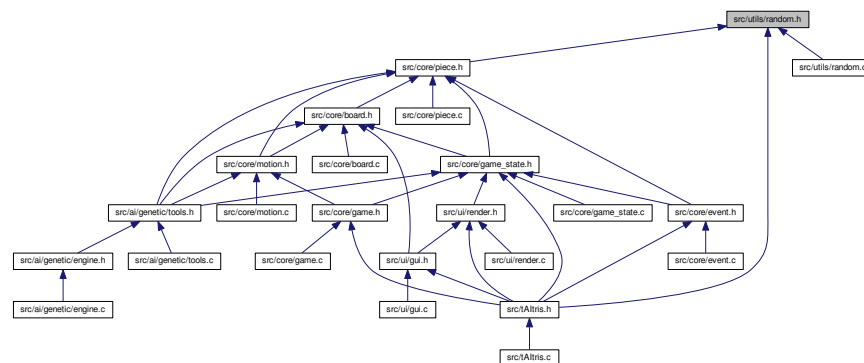
```
#include <stdlib.h>
```

```
#include <time.h>
```

Include dependency graph for random.h:



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



## Functions

- void **random\_init** ()
- size\_t **random\_size\_t** (size\_t min, size\_t max)
- int **random\_int** (int min, int max)

### 4.30.1 Detailed Description

No description.

Author

S4MasterRace

Version

1.0

### 4.30.2 Function Documentation

#### 4.30.2.1 random\_init()

```
void random_init ( ) [inline]
```

#### 4.30.2.2 random\_int()

```
int random_int (
    int min,
    int max ) [inline]
```

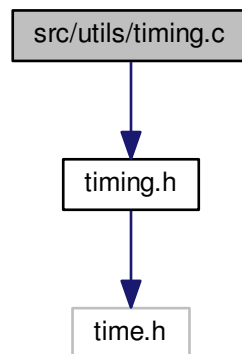
#### 4.30.2.3 random\_size\_t()

```
size_t random_size_t (  
    size_t min,  
    size_t max ) [inline]
```

### 4.31 src/utls/timing.c File Reference

No description.

```
#include "timing.h"  
Include dependency graph for timing.c:
```



#### Functions

- double **time\_get\_current** ()

#### 4.31.1 Detailed Description

No description.

##### Author

S4MasterRace

##### Version

1.0

### 4.31.2 Function Documentation

#### 4.31.2.1 time\_get\_current()

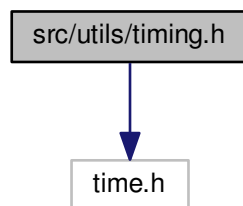
```
double time_get_current ( ) [inline]
```

## 4.32 src/utils/timing.h File Reference

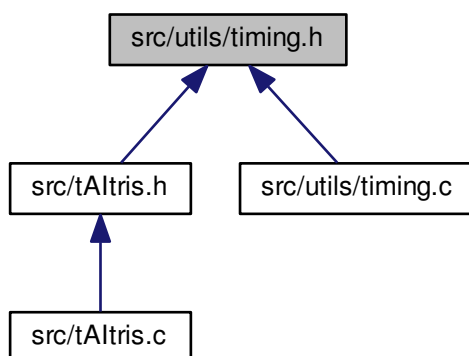
No description.

```
#include <time.h>
```

Include dependency graph for timing.h:



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



## Functions

- double **time\_get\_current** ()

### 4.32.1 Detailed Description

No description.

#### Author

S4MasterRace

#### Version

1.0

### 4.32.2 Function Documentation

#### 4.32.2.1 time\_get\_current()

```
double time_get_current ( ) [inline]
```



# Index

## ABS

tools.h, 20

## agg\_height

ai\_coefs, 5

## aggregate\_height

tools.c, 17

tools.h, 20

## ai\_coefs, 5

agg\_height, 5

bumpiness, 5

clears, 5

holes, 5

## angle

piece, 12

## BOARD\_CELL\_EMPTY

board.h, 27

## BOARD\_HEIGHT

board.h, 27

## BOARD\_WIDTH

board.h, 28

## board, 6

data, 6

game\_state, 7

## board.c

board\_at, 23

board\_break\_line, 23

board\_break\_lines, 23

board\_check\_position, 24

board\_copy, 24

board\_create, 24

board\_free, 24

board\_get\_completed\_lines, 24

board\_init, 24

board\_is\_line\_complete, 24

board\_merge\_piece, 25

board\_move\_line, 25

board\_print, 25

board\_remove\_line, 25

board\_set, 25

## board.h

BOARD\_CELL\_EMPTY, 27

BOARD\_HEIGHT, 27

BOARD\_WIDTH, 28

board\_at, 28

board\_break\_line, 28

board\_break\_lines, 28

board\_check\_position, 28

board\_copy, 28

board\_create, 29

board\_free, 29

board\_get\_completed\_lines, 29

board\_init, 29

board\_is\_line\_complete, 29

board\_merge\_piece, 29

board\_move\_line, 29

board\_print, 30

board\_remove\_line, 30

board\_set, 30

## board\_at

board.c, 23

board.h, 28

## board\_break\_line

board.c, 23

board.h, 28

## board\_break\_lines

board.c, 23

board.h, 28

## board\_check\_position

board.c, 24

board.h, 28

## board\_copy

board.c, 24

board.h, 28

## board\_create

board.c, 24

board.h, 29

## board\_free

board.c, 24

board.h, 29

## board\_get\_completed\_lines

board.c, 24

board.h, 29

## board\_height

tools.c, 17

tools.h, 21

## board\_heights

tools.c, 18

tools.h, 21

## board\_init

board.c, 24

board.h, 29

## board\_is\_line\_complete

board.c, 24

board.h, 29

## board\_merge\_piece

board.c, 25

board.h, 29

## board\_move\_line

- board.c, 25
- board.h, 29
- board\_print
  - board.c, 25
  - board.h, 30
- board\_remove\_line
  - board.c, 25
  - board.h, 30
- board\_set
  - board.c, 25
  - board.h, 30
- broken\_lines
  - game\_state, 7
- bumpiness
  - ai\_coefs, 5
  - tools.c, 18
  - tools.h, 21
- clears
  - ai\_coefs, 5
  - tools.c, 18
  - tools.h, 21
- coalescent\_clears
  - tools.h, 21
- cols
  - matrix, 11
- data
  - board, 6
  - matrix, 12
- down
  - input, 8
- event.c
  - event\_handle, 31
- event.h
  - event\_handle, 33
- event\_handle
  - event.c, 31
  - event.h, 33
- first
  - list, 10
- GS\_SPAWN\_X
  - game\_state.h, 39
- GS\_SPAWN\_Y
  - game\_state.h, 40
- GS\_STATE\_GAMEOVER
  - game\_state.h, 40
- GS\_STATE\_PAUSED
  - game\_state.h, 40
- GS\_STATE\_PLAYING
  - game\_state.h, 40
- GS\_STATE\_QUIT
  - game\_state.h, 40
- GUI\_HEIGHT
  - gui.h, 58
- GUI\_TITLE
  - gui.h, 59
- GUI\_WIDTH
  - gui.h, 59
- game.c
  - game\_tick, 34
- game.h
  - game\_tick, 36
- game\_state, 6
  - board, 7
  - broken\_lines, 7
  - level, 7
  - piece\_current, 7
  - piece\_next, 7
  - score, 7
  - state, 8
  - time, 8
- game\_state.c
  - gs\_create, 37
  - gs\_free, 38
  - gs\_init, 38
  - gs\_next\_piece, 38
- game\_state.h
  - GS\_SPAWN\_X, 39
  - GS\_SPAWN\_Y, 40
  - GS\_STATE\_GAMEOVER, 40
  - GS\_STATE\_PAUSED, 40
  - GS\_STATE\_PLAYING, 40
  - GS\_STATE\_QUIT, 40
  - gs\_create, 40
  - gs\_free, 40
  - gs\_init, 41
  - gs\_next\_piece, 41
- game\_tick
  - game.c, 34
  - game.h, 36
- gs\_create
  - game\_state.c, 37
  - game\_state.h, 40
- gs\_free
  - game\_state.c, 38
  - game\_state.h, 40
- gs\_init
  - game\_state.c, 38
  - game\_state.h, 41
- gs\_next\_piece
  - game\_state.c, 38
  - game\_state.h, 41
- gui.c
  - gui\_free, 57
  - gui\_init, 57
  - gui\_load\_image, 57
- gui.h
  - GUI\_HEIGHT, 58
  - GUI\_TITLE, 59
  - GUI\_WIDTH, 59
  - gui\_free, 59
  - gui\_init, 59
  - gui\_load\_image, 59

- gui\_free
  - gui.c, 57
  - gui.h, 59
- gui\_init
  - gui.c, 57
  - gui.h, 59
- gui\_load\_image
  - gui.c, 57
  - gui.h, 59
- hole
  - tools.c, 18
  - tools.h, 21
- holes
  - ai\_coefs, 5
  - tools.c, 18
  - tools.h, 22
- id
  - piece, 12
- input, 8
  - down, 8
  - moveX, 8
  - moveY, 9
  - quit, 9
  - rotate, 9
- length
  - list, 10
- level
  - game\_state, 7
- list, 9
  - first, 10
  - length, 10
- list.c
  - list\_add, 65
  - list\_advance, 66
  - list\_append, 66
  - list\_at, 68
  - list\_concat, 68
  - list\_del, 69
  - list\_del\_after, 70
  - list\_del\_at, 70
  - list\_first, 71
  - list\_init, 71
  - list\_insert\_after, 72
  - list\_insert\_at, 72
  - list\_is\_empty, 73
  - list\_last, 74
  - list\_length, 74
  - list\_next, 75
  - list\_print, 75
  - list\_reverse, 75
  - list\_sort, 77
  - list\_split\_at, 77
  - list\_swap, 78
- list.h
  - list\_add, 84
  - list\_advance, 84
  - list\_append, 85
  - list\_at, 85
  - list\_concat, 87
  - list\_del, 87
  - list\_del\_after, 88
  - list\_del\_at, 89
  - list\_elt, 81
  - list\_first, 89
  - list\_foreach, 81
  - list\_foreach\_elt, 82
  - list\_foreach\_elt\_safe, 82
  - list\_foreach\_safe, 83
  - list\_init, 90
  - list\_insert\_after, 90
  - list\_insert\_at, 91
  - list\_is\_empty, 91
  - list\_last, 92
  - list\_length, 92
  - list\_next, 93
  - list\_print, 93
  - list\_reverse, 94
  - list\_sort, 94
  - list\_split\_at, 95
  - list\_swap, 96
- list\_add
  - list.c, 65
  - list.h, 84
- list\_advance
  - list.c, 66
  - list.h, 84
- list\_append
  - list.c, 66
  - list.h, 85
- list\_at
  - list.c, 68
  - list.h, 85
- list\_concat
  - list.c, 68
  - list.h, 87
- list\_del
  - list.c, 69
  - list.h, 87
- list\_del\_after
  - list.c, 70
  - list.h, 88
- list\_del\_at
  - list.c, 70
  - list.h, 89
- list\_elt
  - list.h, 81
- list\_first
  - list.c, 71
  - list.h, 89
- list\_foreach
  - list.h, 81
- list\_foreach\_elt
  - list.h, 82
- list\_foreach\_elt\_safe

- list.h, 82
- list\_foreach\_safe
  - list.h, 83
- list\_init
  - list.c, 71
  - list.h, 90
- list\_insert\_after
  - list.c, 72
  - list.h, 90
- list\_insert\_at
  - list.c, 72
  - list.h, 91
- list\_is\_empty
  - list.c, 73
  - list.h, 91
- list\_last
  - list.c, 74
  - list.h, 92
- list\_length
  - list.c, 74
  - list.h, 92
- list\_next
  - list.c, 75
  - list.h, 93
- list\_node, 10
  - next, 11
- list\_print
  - list.c, 75
  - list.h, 93
- list\_reverse
  - list.c, 75
  - list.h, 94
- list\_sort
  - list.c, 77
  - list.h, 94
- list\_split\_at
  - list.c, 77
  - list.h, 95
- list\_swap
  - list.c, 78
  - list.h, 96
- main
  - tAltris.c, 54
- make\_line
  - tools.c, 18
  - tools.h, 22
- matrix, 11
  - cols, 11
  - data, 12
  - rows, 12
- matrix.c
  - matrix\_at, 97
  - matrix\_cols, 98
  - matrix\_copy, 98
  - matrix\_create, 99
  - matrix\_create\_from\_array, 99
  - matrix\_diagonal, 101
  - matrix\_dot\_product, 101
- matrix\_free, 102
- matrix\_hadamard\_product, 103
- matrix\_identity, 103
- matrix\_is\_diagonal, 104
- matrix\_is\_square, 104
- matrix\_is\_upper\_triangularized, 105
- matrix\_print, 105
- matrix\_product, 106
- matrix\_rows, 106
- matrix\_scale, 107
- matrix\_set, 108
- matrix\_sum, 108
- matrix\_transpose, 109
- matrix.h
  - matrix\_at, 111
  - matrix\_cols, 112
  - matrix\_copy, 112
  - matrix\_create, 113
  - matrix\_create\_from\_array, 113
  - matrix\_diagonal, 114
  - matrix\_dot\_product, 114
  - matrix\_free, 115
  - matrix\_hadamard\_product, 116
  - matrix\_identity, 116
  - matrix\_is\_diagonal, 117
  - matrix\_is\_square, 117
  - matrix\_is\_upper\_triangularized, 118
  - matrix\_print, 118
  - matrix\_product, 119
  - matrix\_rows, 119
  - matrix\_scale, 120
  - matrix\_set, 121
  - matrix\_sum, 121
  - matrix\_transpose, 122
- matrix\_at
  - matrix.c, 97
  - matrix.h, 111
- matrix\_cols
  - matrix.c, 98
  - matrix.h, 112
- matrix\_copy
  - matrix.c, 98
  - matrix.h, 112
- matrix\_create
  - matrix.c, 99
  - matrix.h, 113
- matrix\_create\_from\_array
  - matrix.c, 99
  - matrix.h, 113
- matrix\_diagonal
  - matrix.c, 101
  - matrix.h, 114
- matrix\_dot\_product
  - matrix.c, 101
  - matrix.h, 114
- matrix\_free
  - matrix.c, 102
  - matrix.h, 115

- matrix\_hadamard\_product
  - matrix.c, 103
  - matrix.h, 116
- matrix\_identity
  - matrix.c, 103
  - matrix.h, 116
- matrix\_is\_diagonal
  - matrix.c, 104
  - matrix.h, 117
- matrix\_is\_square
  - matrix.c, 104
  - matrix.h, 117
- matrix\_is\_upper\_triangular
  - matrix.c, 105
  - matrix.h, 118
- matrix\_print
  - matrix.c, 105
  - matrix.h, 118
- matrix\_product
  - matrix.c, 106
  - matrix.h, 119
- matrix\_rows
  - matrix.c, 106
  - matrix.h, 119
- matrix\_scale
  - matrix.c, 107
  - matrix.h, 120
- matrix\_set
  - matrix.c, 108
  - matrix.h, 121
- matrix\_sum
  - matrix.c, 108
  - matrix.h, 121
- matrix\_transpose
  - matrix.c, 109
  - matrix.h, 122
- motion.c
  - motion\_can\_move, 43
  - motion\_can\_rotate, 44
  - motion\_try\_move, 44
  - motion\_try\_move\_down, 44
  - motion\_try\_rotate, 44
- motion.h
  - motion\_can\_move, 46
  - motion\_can\_rotate, 46
  - motion\_try\_move, 46
  - motion\_try\_move\_down, 46
  - motion\_try\_rotate, 47
- motion\_can\_move
  - motion.c, 43
  - motion.h, 46
- motion\_can\_rotate
  - motion.c, 44
  - motion.h, 46
- motion\_try\_move
  - motion.c, 44
  - motion.h, 46
- motion\_try\_move\_down
  - motion.c, 44
  - motion.h, 46
- motion\_try\_rotate
  - motion.c, 44
  - motion.h, 46
- motion\_try\_rotate
  - motion.c, 44
  - motion.h, 47
- moveX
  - input, 8
- moveY
  - input, 9
- next
  - list\_node, 11
- PIECE\_ANGLE\_DOWN
  - piece.h, 50
- PIECE\_ANGLE\_LEFT
  - piece.h, 51
- PIECE\_ANGLE\_RIGHT
  - piece.h, 51
- PIECE\_ANGLE\_UP
  - piece.h, 51
- PIECE\_ANGLES
  - piece.h, 51
- PIECE\_COUNT
  - piece.h, 51
- PIECE\_HEIGHT
  - piece.h, 51
- PIECE\_ROTATE\_LEFT
  - piece.h, 52
- PIECE\_ROTATE\_RIGHT
  - piece.h, 52
- PIECE\_SHAPES
  - piece.c, 48
  - piece.h, 53
- PIECE\_WIDTH
  - piece.h, 52
- PIECE\_I
  - piece.h, 51
- PIECE\_J
  - piece.h, 51
- PIECE\_L
  - piece.h, 52
- PIECE\_O
  - piece.h, 52
- PIECE\_S
  - piece.h, 52
- PIECE\_T
  - piece.h, 52
- PIECE\_Z
  - piece.h, 52
- piece, 12
  - angle, 12
  - id, 12
  - shapes, 13
  - x, 13
  - y, 13
- piece.c
  - PIECE\_SHAPES, 48
  - piece\_move, 48

- piece\_random, 48
  - piece\_rotate, 48
- piece.h
  - PIECE\_ANGLE\_DOWN, 50
  - PIECE\_ANGLE\_LEFT, 51
  - PIECE\_ANGLE\_RIGHT, 51
  - PIECE\_ANGLE\_UP, 51
  - PIECE\_ANGLES, 51
  - PIECE\_COUNT, 51
  - PIECE\_HEIGHT, 51
  - PIECE\_ROTATE\_LEFT, 52
  - PIECE\_ROTATE\_RIGHT, 52
  - PIECE\_SHAPES, 53
  - PIECE\_WIDTH, 52
  - PIECE\_I, 51
  - PIECE\_J, 51
  - PIECE\_L, 52
  - PIECE\_O, 52
  - PIECE\_S, 52
  - PIECE\_T, 52
  - PIECE\_Z, 52
  - piece\_move, 53
  - piece\_random, 53
  - piece\_rotate, 53
- piece\_current
  - game\_state, 7
- piece\_move
  - piece.c, 48
  - piece.h, 53
- piece\_next
  - game\_state, 7
- piece\_random
  - piece.c, 48
  - piece.h, 53
- piece\_rotate
  - piece.c, 48
  - piece.h, 53
- quit
  - input, 9
- RENDER\_CELL\_SIZE
  - render.h, 63
- RENDER\_FPS
  - render.h, 63
- random.c
  - random\_init, 123
  - random\_int, 124
  - random\_size\_t, 124
- random.h
  - random\_init, 125
  - random\_int, 125
  - random\_size\_t, 125
- random\_init
  - random.c, 123
  - random.h, 125
- random\_int
  - random.c, 124
  - random.h, 125
- random\_size\_t
  - random.c, 124
  - random.h, 125
- render.c
  - render\_board, 61
  - render\_handle, 61
  - render\_next\_piece, 61
  - render\_piece, 61
- render.h
  - RENDER\_CELL\_SIZE, 63
  - RENDER\_FPS, 63
  - render\_board, 63
  - render\_handle, 63
  - render\_next\_piece, 64
  - render\_piece, 64
- render\_board
  - render.c, 61
  - render.h, 63
- render\_handle
  - render.c, 61
  - render.h, 63
- render\_next\_piece
  - render.c, 61
  - render.h, 64
- render\_piece
  - render.c, 61
  - render.h, 64
- rotate
  - input, 9
- rows
  - matrix, 12
- score
  - game\_state, 7
- shapes
  - piece, 13
- src/ai/genetic/engine.c, 15
- src/ai/genetic/engine.h, 16
- src/ai/genetic/tools.c, 17
- src/ai/genetic/tools.h, 19
- src/core/board.c, 22
- src/core/board.h, 26
- src/core/event.c, 30
- src/core/event.h, 32
- src/core/game.c, 34
- src/core/game.h, 35
- src/core/game\_state.c, 37
- src/core/game\_state.h, 38
- src/core/input.c, 41
- src/core/input.h, 42
- src/core/motion.c, 42
- src/core/motion.h, 45
- src/core/piece.c, 47
- src/core/piece.h, 49
- src/tAltris.c, 54
- src/tAltris.h, 55
- src/ui/gui.c, 56
- src/ui/gui.h, 57
- src/ui/render.c, 60

- src/ui/render.h, 61
- src/utls/list.c, 64
- src/utls/list.h, 79
- src/utls/matrix.c, 96
- src/utls/matrix.h, 110
- src/utls/random.c, 123
- src/utls/random.h, 124
- src/utls/timing.c, 126
- src/utls/timing.h, 127
- state
  - game\_state, 8
- tAltris.c
  - main, 54
- time
  - game\_state, 8
- time\_get\_current
  - timing.c, 127
  - timing.h, 128
- timing.c
  - time\_get\_current, 127
- timing.h
  - time\_get\_current, 128
- tools.c
  - aggregate\_height, 17
  - board\_height, 17
  - board\_heights, 18
  - bumpiness, 18
  - clears, 18
  - hole, 18
  - holes, 18
  - make\_line, 18
- tools.h
  - ABS, 20
  - aggregate\_height, 20
  - board\_height, 21
  - board\_heights, 21
  - bumpiness, 21
  - clears, 21
  - coalescent\_clears, 21
  - hole, 21
  - holes, 22
  - make\_line, 22
- x
  - piece, 13
- y
  - piece, 13