tAltris

v1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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]
```

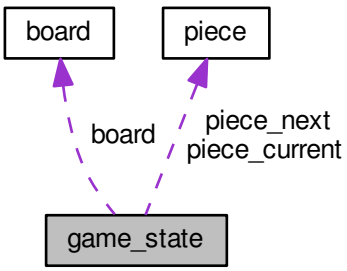
The documentation for this struct was generated from the following file:

- 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
```

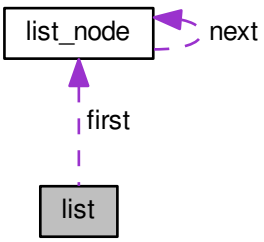
The documentation for this struct was generated from the following file:

- 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.

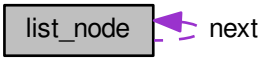
The documentation for this struct was generated from the following file:

- 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/utils/ **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**

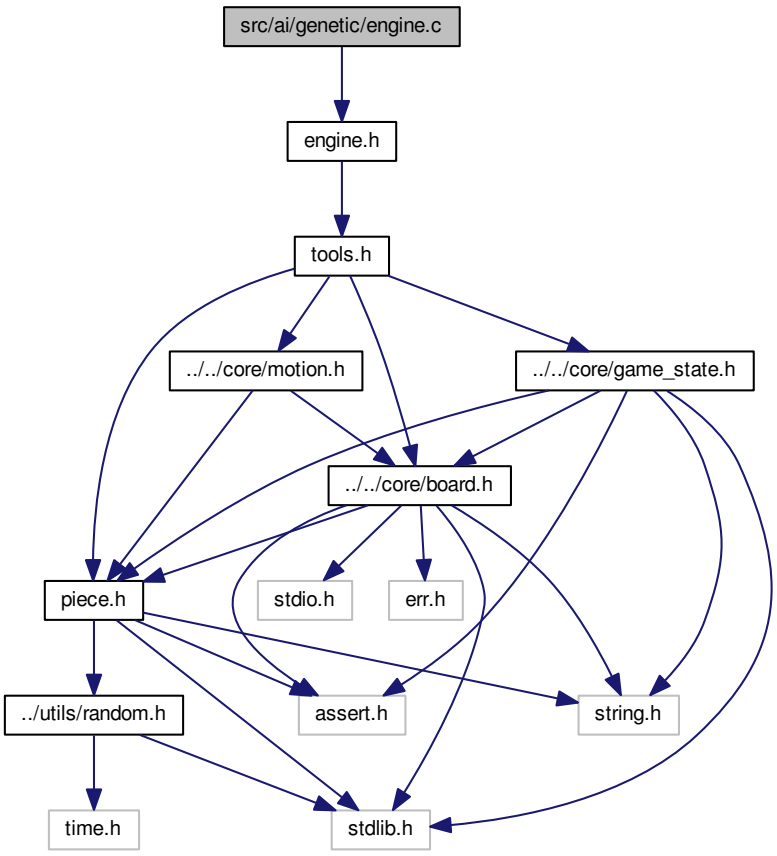
Chapter 4

File Documentation

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

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

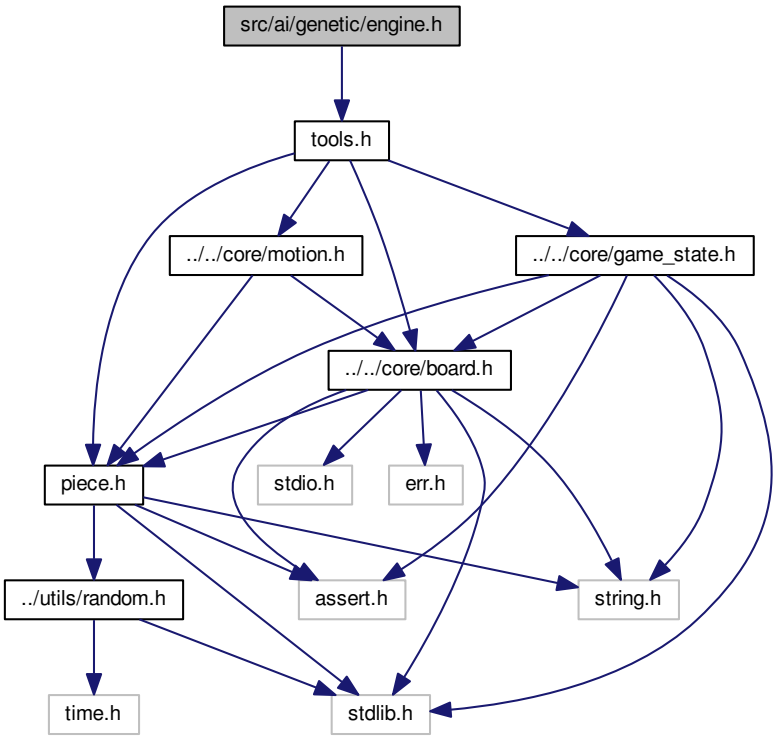
Include dependency graph for engine.c:



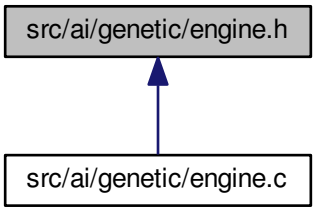
4.2 src/ai/genetic/engine.h File Reference

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

Include dependency graph for engine.h:



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



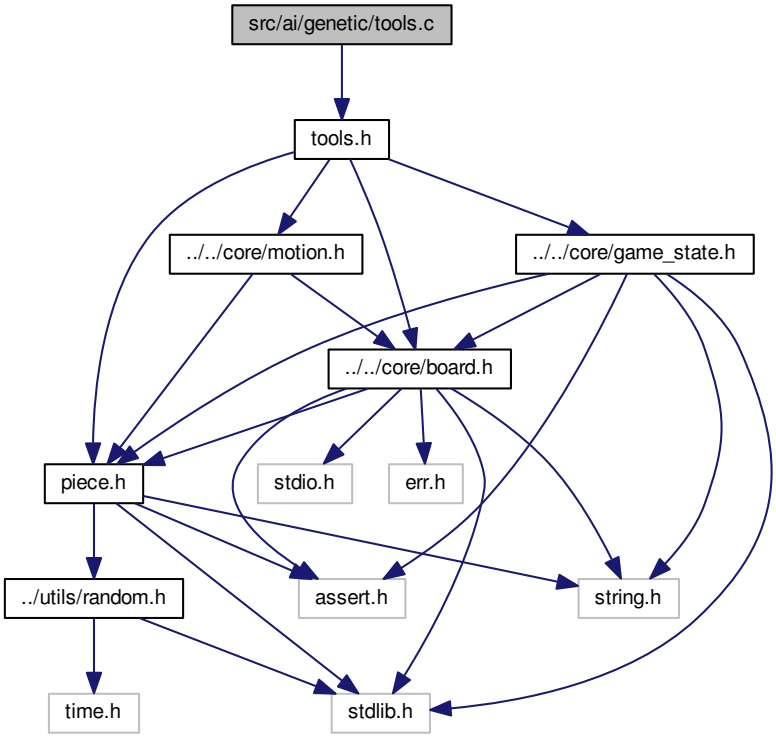
Data Structures

- 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]
```

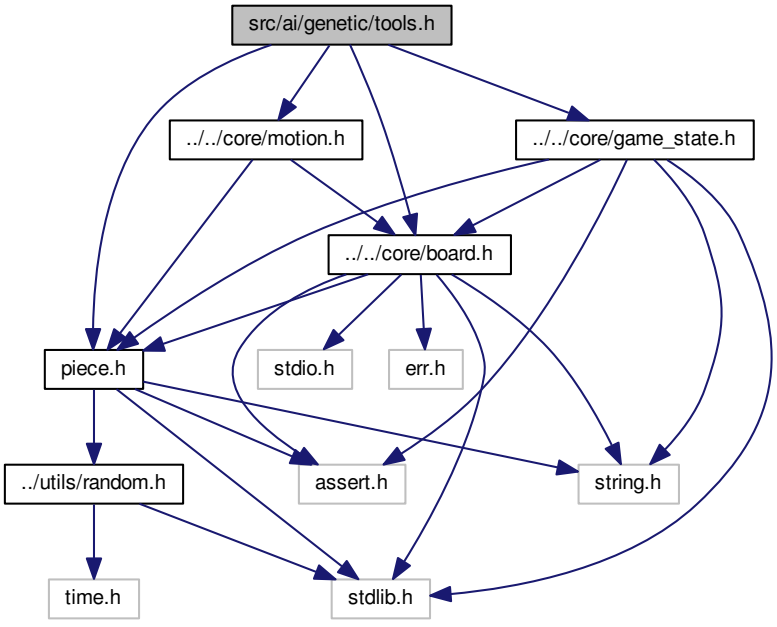
4.3.1.8 make_line()

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

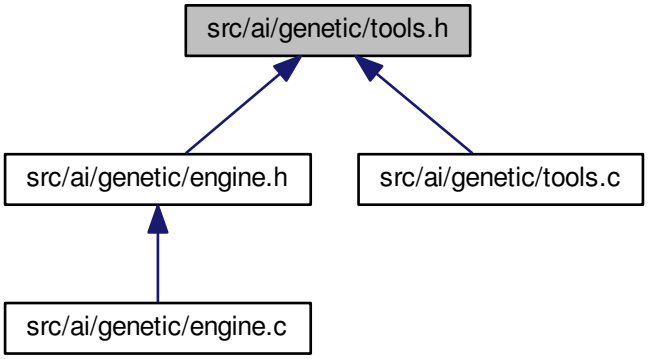
4.4 src/ai/genetic/tools.h File Reference

```
#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()

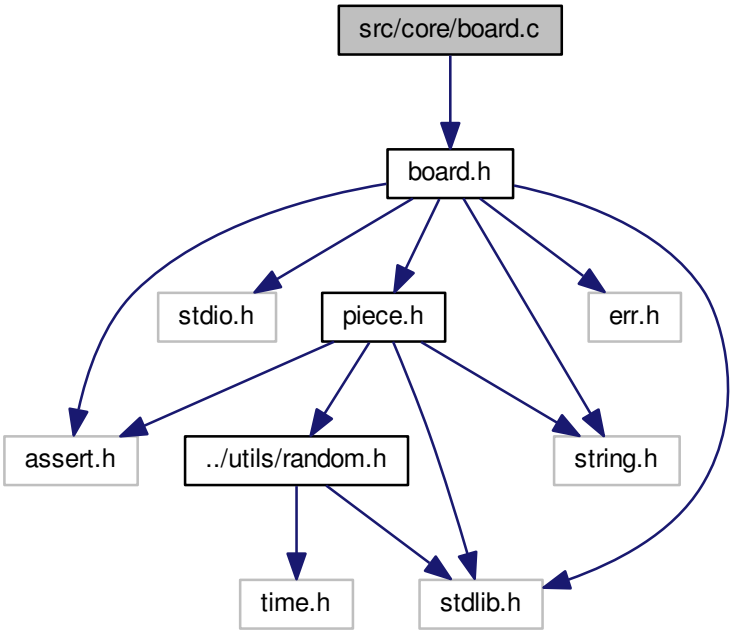
```
void make_line (
            struct  board * brd,
            size_t y )  [inline]
```

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()

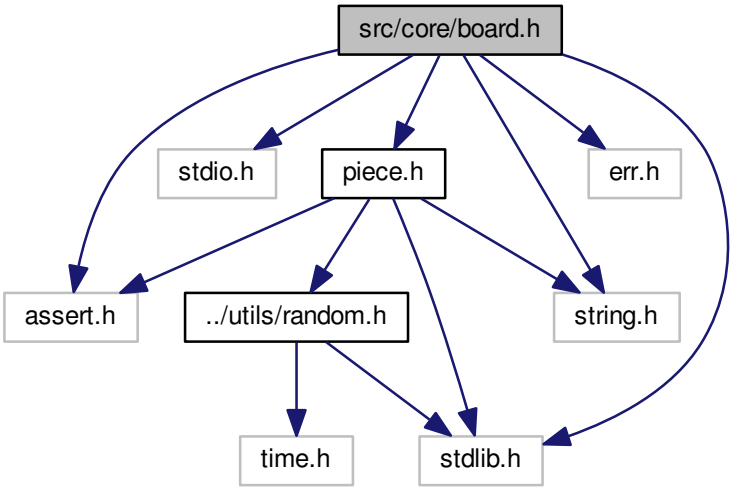
```
void board_set (
            struct  board * brd,
            size_t x,
            size_t y,
            int state )  [inline]
```

4.6 src/core/board.h File Reference

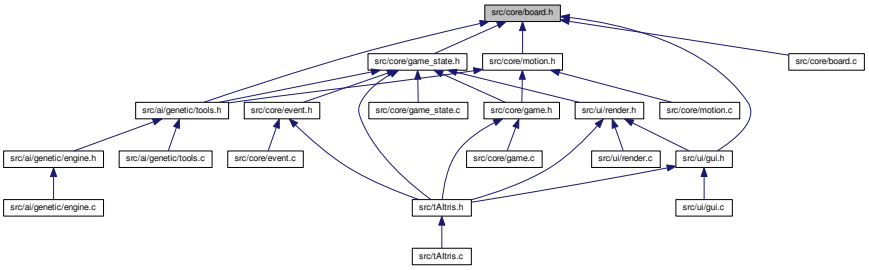
No description.

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

Include dependency graph for board.h:



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



Data Structures

- struct **board**

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()

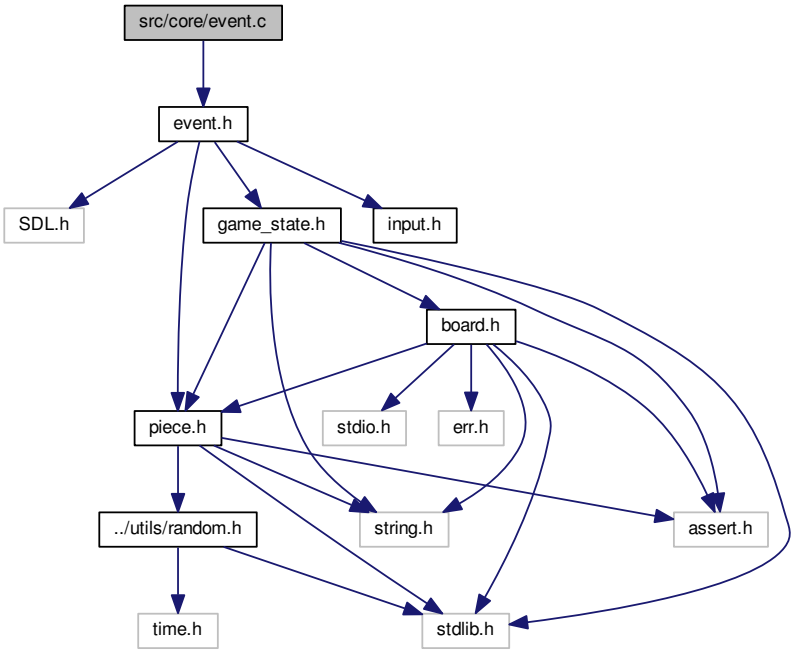
```
void board_set (
            struct  board * brd,
            size_t x,
            size_t y,
            int state )  [inline]
```

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()

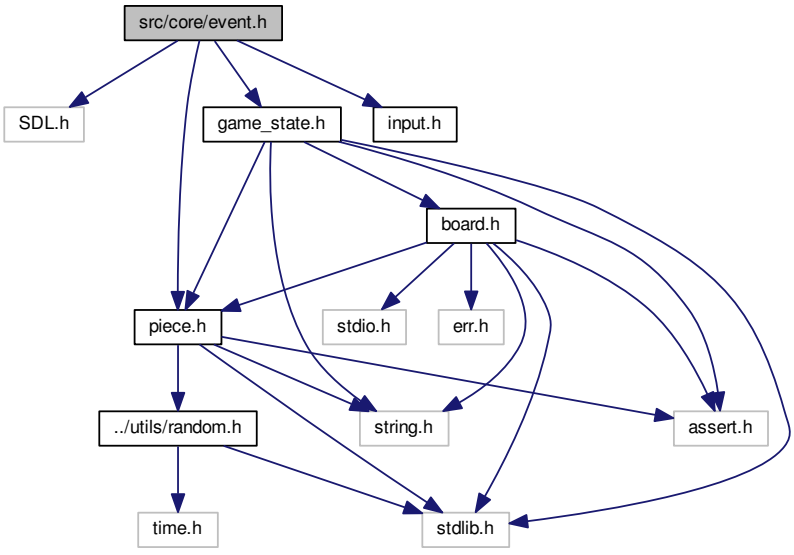
```
void event_handle (
            struct  game_state * gs,
            struct  input * in )
```

4.8 src/core/event.h File Reference

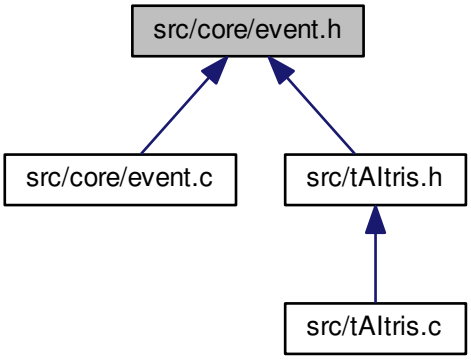
No description.

```
#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()

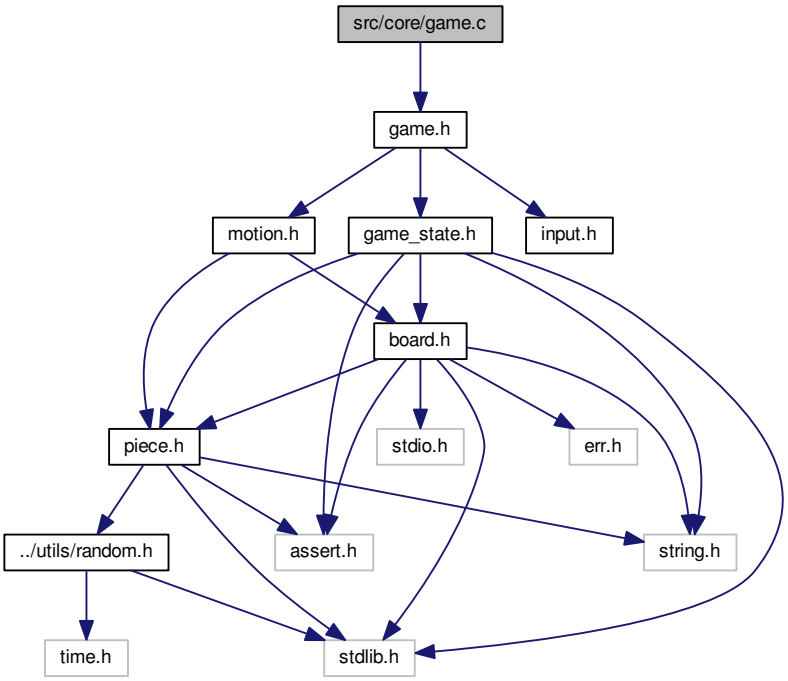
```
void event_handle (
            struct  game_state * gs,
            struct  input * in )
```

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()

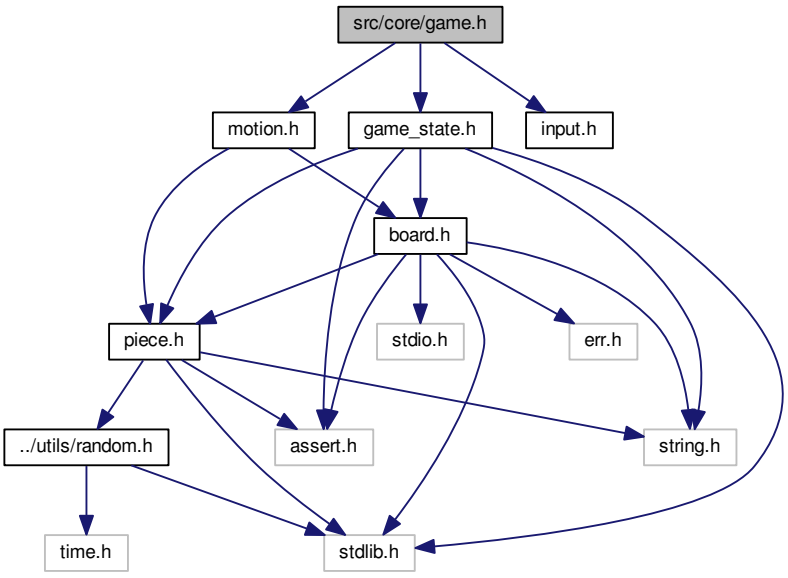
```
void game_tick (
            double dt,
            struct  game_state * gs,
            struct  input * in )
```

4.10 src/core/game.h File Reference

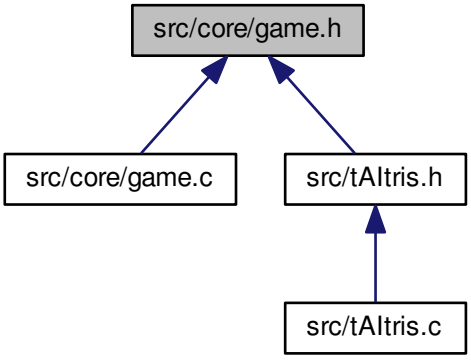
No description.

```
#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()

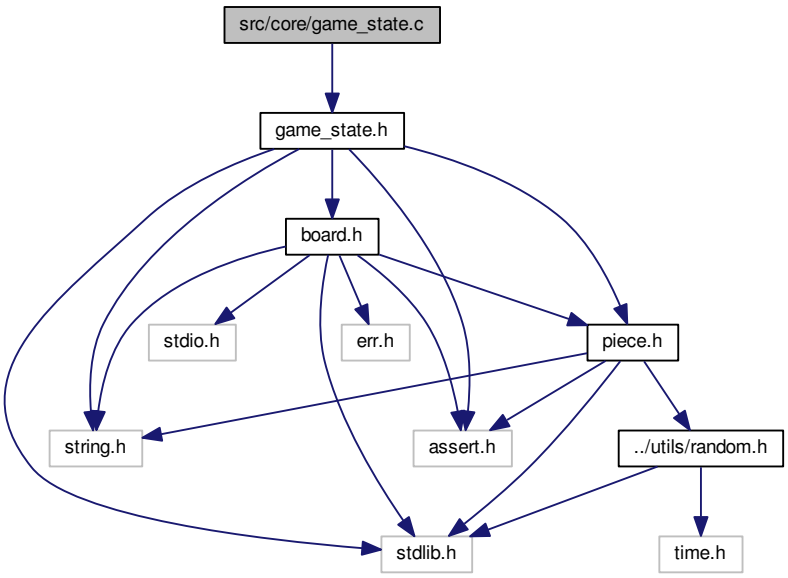
```
void game_tick (
            double dt,
            struct  game_state * gs,
            struct  input * in )
```

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()

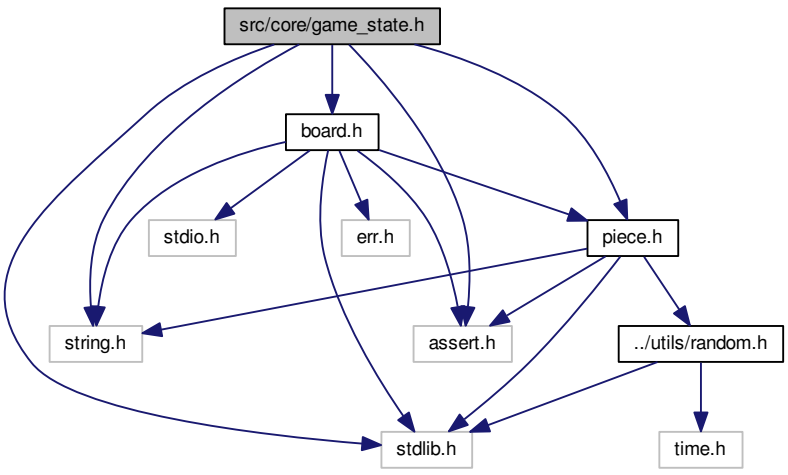
```
int gs_next_piece (
            struct  game_state * gs )  [inline]
```

4.12 src/core/game_state.h File Reference

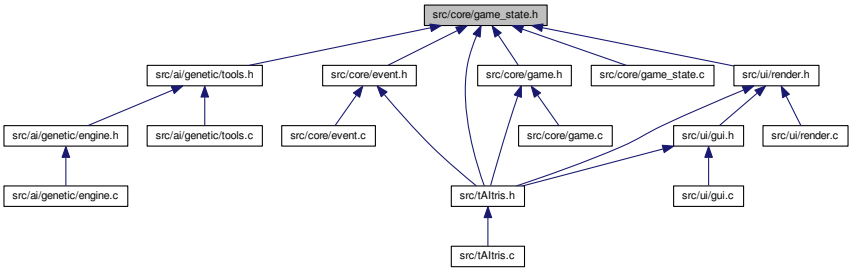
No description.

```
#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()

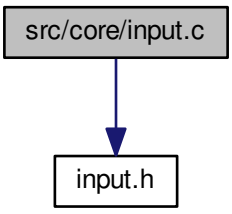
```
int gs_next_piece (
            struct  game_state * gs )  [inline]
```

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

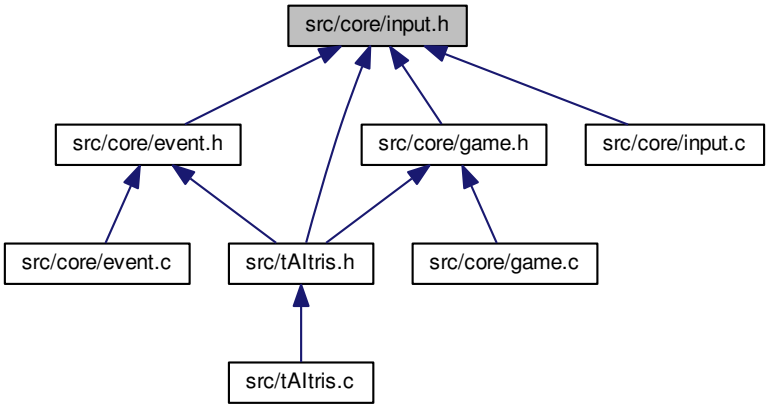
Version

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

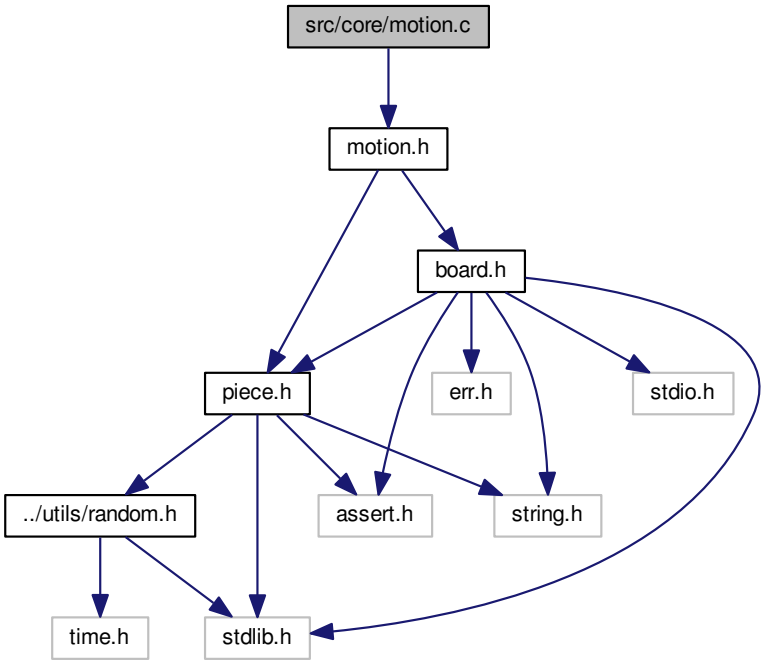
1.0

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()

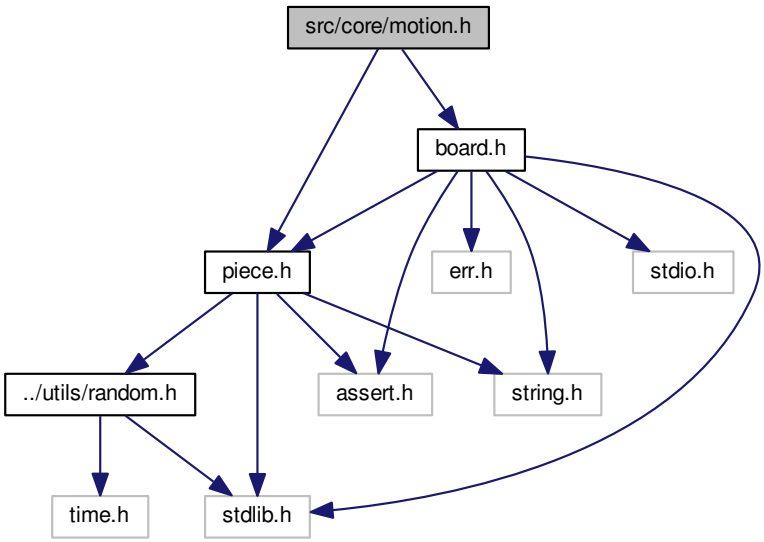
```
int motion_try_rotate (
            struct  piece * pc,
            const struct  board * brd,
            int rotation )
```

4.16 src/core/motion.h File Reference

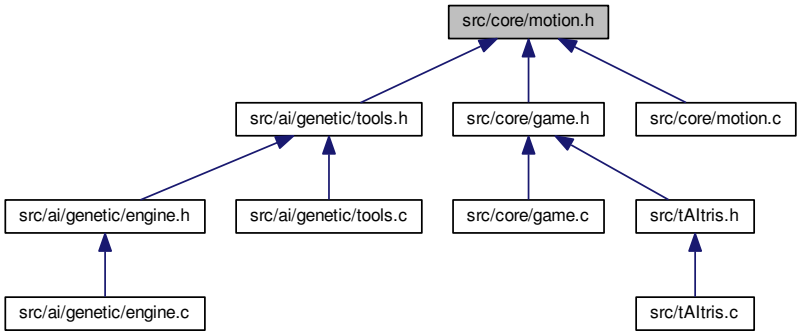
No description.

```
#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()

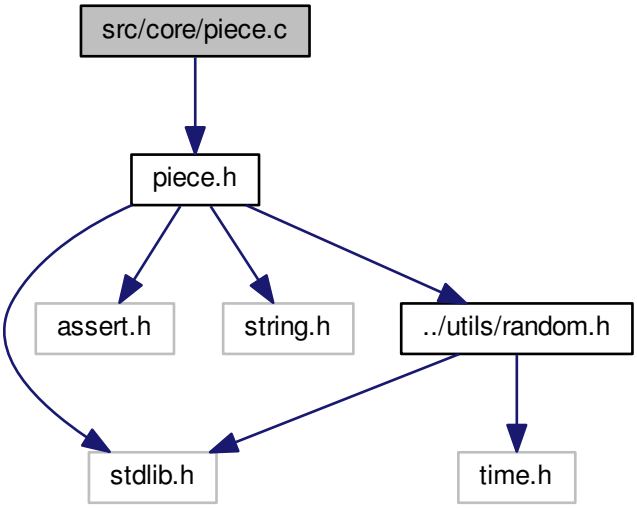
```
int motion_try_rotate (
            struct  piece * pc,
            const struct  board * brd,
            int rotation )
```

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

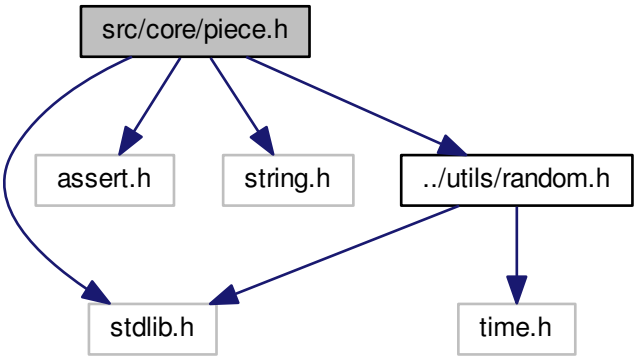
```
const int PIECE_SHAPES[7][4][4][4]
```

4.18 src/core/piece.h File Reference

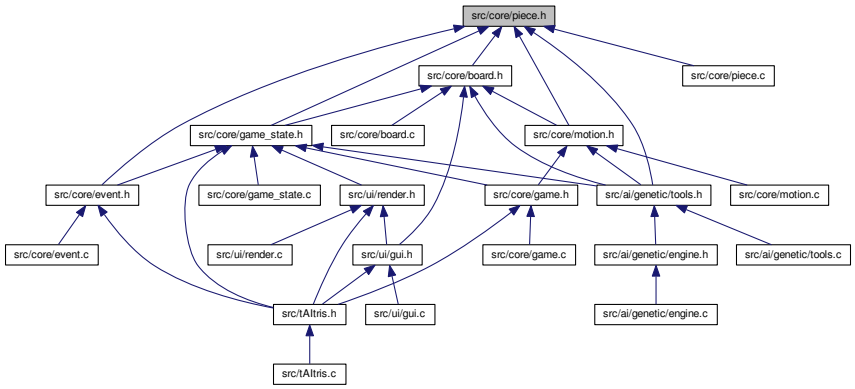
No description.

```
#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

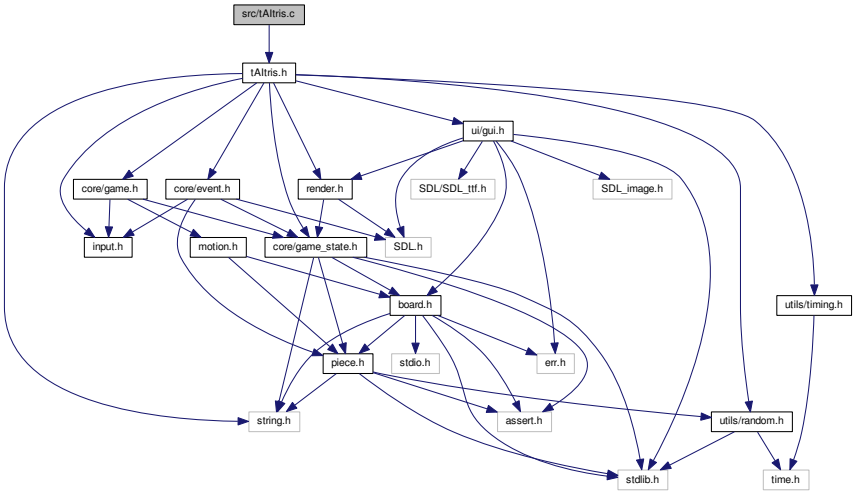
```
const int PIECE_SHAPES[ PIECE_COUNT][ PIECE_ANGLES][ PIECE_HEIGHT][ PIECE_WIDTH]
```

4.19 src/tAItris.c File Reference

Main file.

```
#include "tAItris.h"
```

Include dependency graph for tAItris.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()

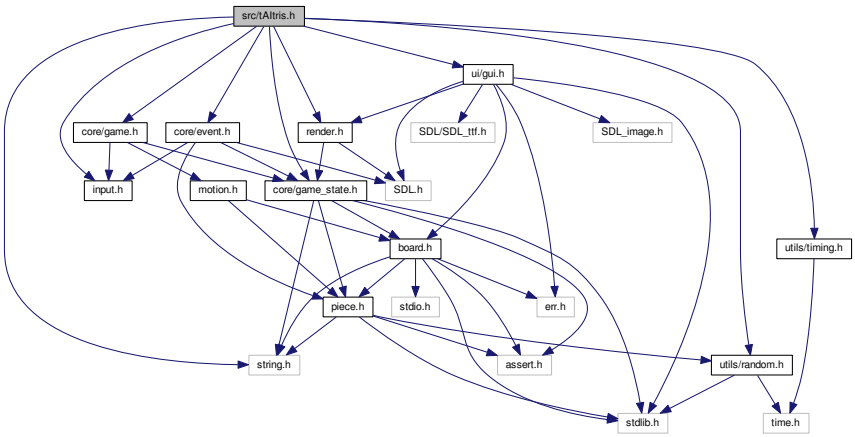
```
int main ( )
```

4.20 src/tAltris.h File Reference

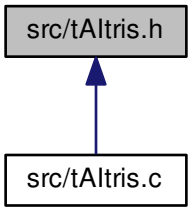
Main file.

```
#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

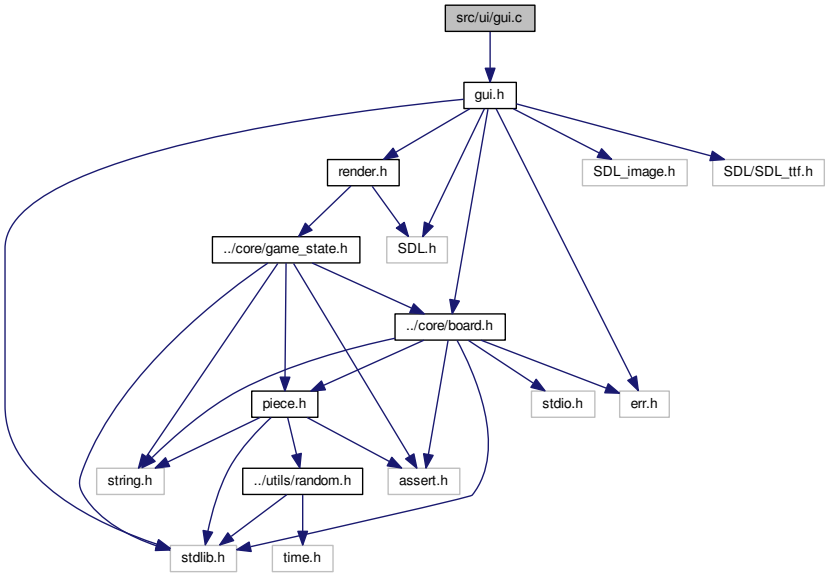
1.0

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()

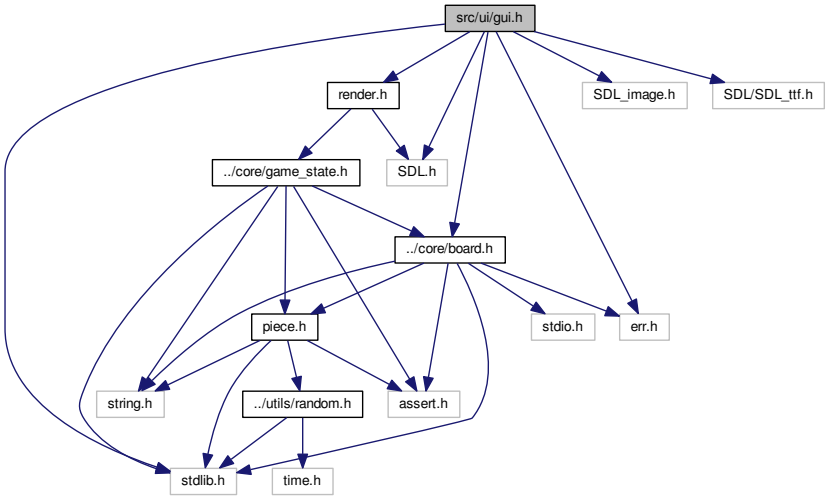
```
SDL_Surface* gui_load_image (
            char * path )  [inline]
```

4.22 src/ui/gui.h File Reference

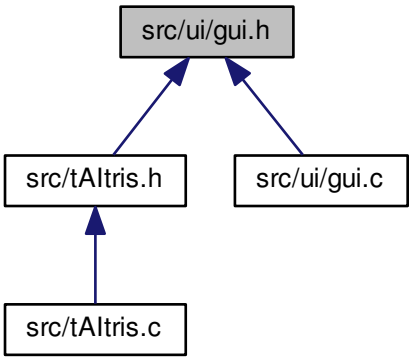
No description.

```
#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** "tAItris"
- #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 "tAItris"
```

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()

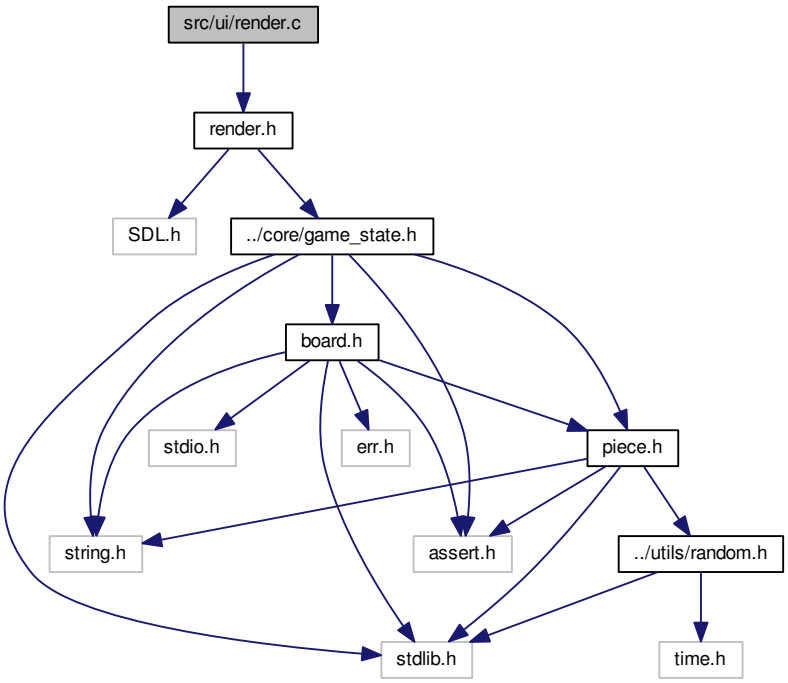
```
SDL_Surface* gui_load_image (
            char * path )  [inline]
```

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()

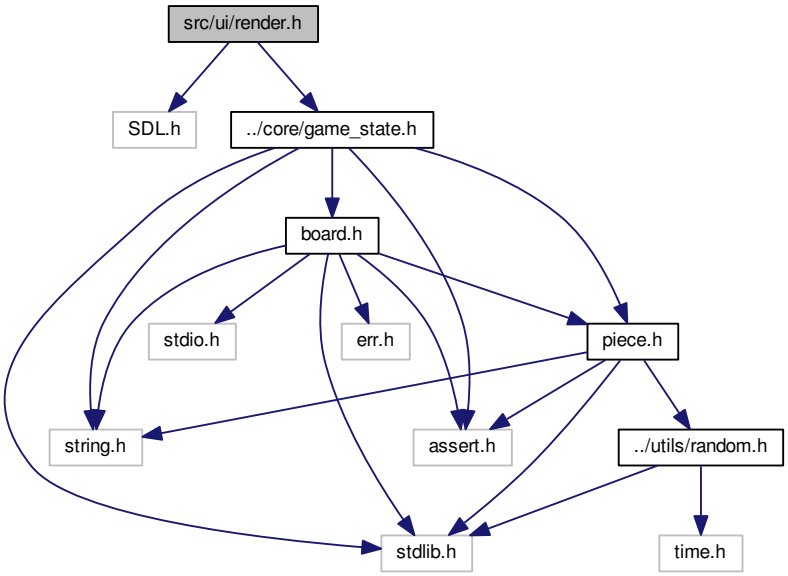
```
void render_piece (
            SDL_Surface * screen,
            struct  piece pc )
```

4.24 src/ui/render.h File Reference

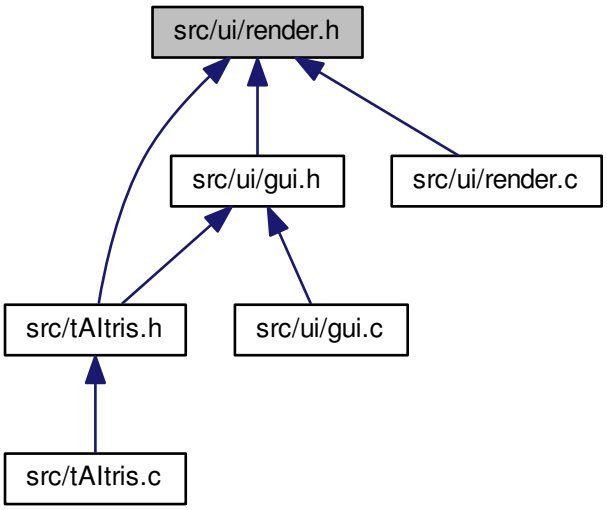
No description.

```
#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()

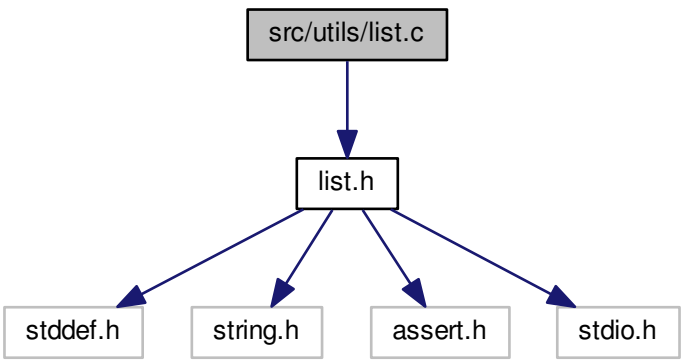
```
void render_piece (
            SDL_Surface * screen,
            struct  piece pc )
```

4.25 src/utils/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

list	a list.
node	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

node	a node.
distance	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

list	a list.
node	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

list	a list.
pos	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

l1	list 1.
l2	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

list	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

list	a list.
node	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

list	a list.
pos	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

list	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

list	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

list	a list.
curr	a node of `list`.
node	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

list	a list.
node	new node.
pos	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

list	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

list	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

list	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

node	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

list	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

list	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

list	list to sort.
cmp	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

list	list to split.
pos	position (0-based) where to split the list.
right	an empty list to receive the part after `pos`

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

l1	list 1.
l2	list 2.

Precondition

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

Remarks

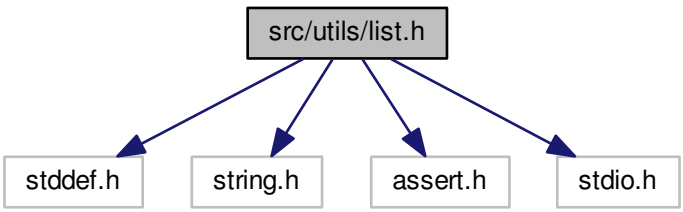
Complexity: O(1)

4.26 src/utils/list.h File Reference

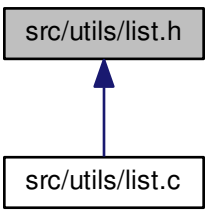
Intrusive list implement.

```
#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

node	a list node (struct list_node*).
type	type of the structure which contains the node.
fieldname	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

list	a list (struct list*).
curr	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

list	a list (struct list∗).
curr	pointer (type∗) used to hold the current element.
type	type of the structure which contains the node.
fieldname	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

list	a list (struct list∗).
curr	pointer (type∗) used to hold the current element.
tmp	a struct list_node∗ used as temporary storage.
type	type of the structure which contains the node.
fieldname	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

list	a list (struct list∗).
curr	a struct list_node∗ used to hold the current element.
tmp	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

list	a list.
node	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 nth-node after the current one.

Parameters

node	a node.
distance	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

list	a list.
node	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

list	a list.
pos	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

l1	list 1.
l2	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

list	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

list	a list.
node	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)

4.26.3.8 list_del_at()

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

Deletes the node at the position `pos`.

Parameters

list	a list.
pos	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)

4.26.3.9 list_first()

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

Returns the first node.

Parameters

list	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

list	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

list	a list.
curr	a node of `list`.
node	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

list	a list.
node	new node.
pos	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

list	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

list	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

list	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

node	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

list	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

list	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

list	list to sort.
cmp	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

list	list to split.
pos	position (0-based) where to split the list.
right	an empty list to receive the part after `pos`

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

l1	list 1.
l2	list 2.

Precondition

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

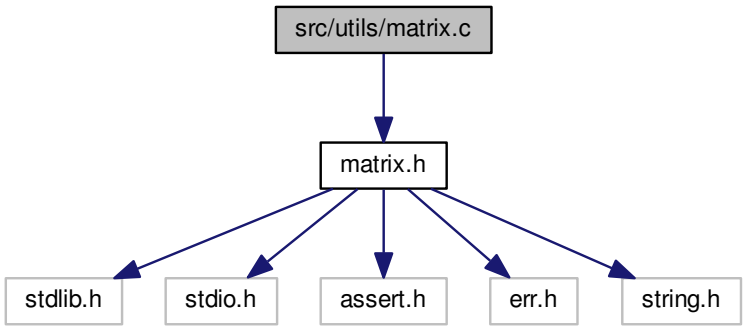
Remarks

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_triangulared** (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

mx	a matrix
rows	rows
cols	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

mx	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

mx	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

rows	number of rows
cols	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

rows	number of rows
cols	number of columns
values	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

v	a vector
mx	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 equel to `matrix_rows(mx)`
`matrix_rows(mx)` must be equel 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

v1	a vector
v2	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

mx	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

mx1	a matrix
mx2	a matrix
prod	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

mx	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

mx	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

mx	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_triangulared()

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

Check if the matrix `mx` is upper triangulared

Parameters

mx	a matrix

Returns

`1` if the matrix is upper triangulared, `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

mx	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

mx1	a matrix
mx2	a matrix
prod	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

mx	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

mx	a matrix
scale	the scale factor
smx	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

mx	a matrix
rows	rows
cols	columns
value	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

mx1	a matrix
mx2	a matrix
sum	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

mx	a matrix
tmx	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

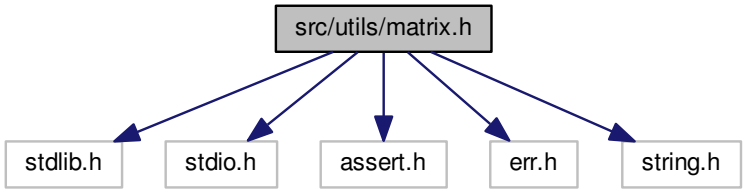
Complexity: O(N)

4.28 src/utils/matrix.h File Reference

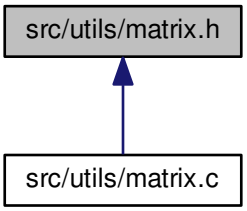
Matrix implement.

```
#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_triangulared** (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

mx	a matrix
rows	rows
cols	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

mx	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

mx	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

rows	number of rows
cols	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

rows	number of rows
cols	number of columns
values	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

v	a vector
mx	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 equel to `matrix_rows(mx)`
`matrix_rows(mx)` must be equel 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

v1	a vector
v2	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

mx	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

mx1	a matrix
mx2	a matrix
prod	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

mx	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

mx	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

mx	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_triangulared()

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

Check if the matrix `mx` is upper triangulared

Parameters

mx	a matrix

Returns

`1` if the matrix is upper triangulared, `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

mx	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

mx1	a matrix
mx2	a matrix
prod	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

mx	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

mx	a matrix
scale	the scale factor
smx	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

mx	a matrix
rows	rows
cols	columns
value	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

mx1	a matrix
mx2	a matrix
sum	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

mx	a matrix
tmx	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

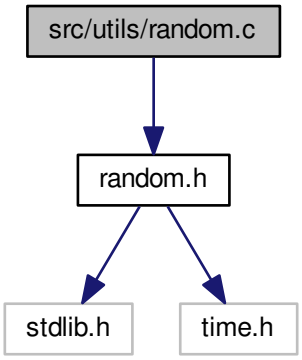
Complexity: O(N)

4.29 src/utils/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()

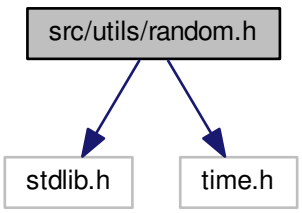
```
size_t random_size_t (
            size_t min,
            size_t max )  [inline]
```

4.30 src/utils/random.h File Reference

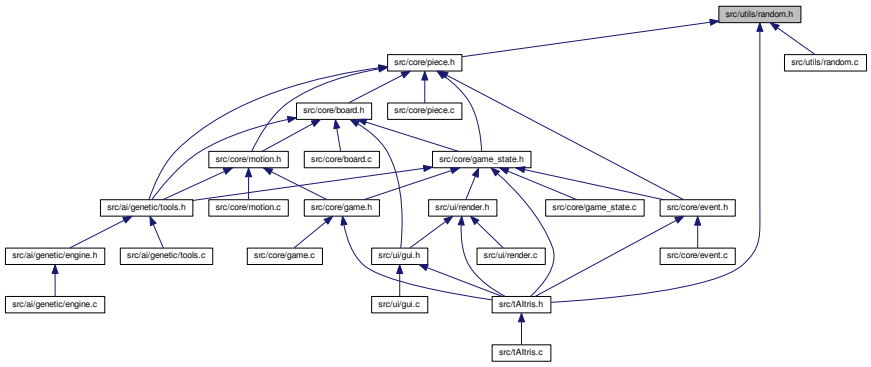
No description.

```
#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()

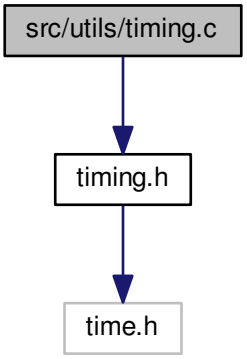
```
size_t random_size_t (
            size_t min,
            size_t max )  [inline]
```

4.31 src/utils/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()

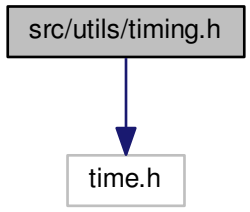
```
double time_get_current ( )  [inline]
```

4.32 src/utils/timing.h File Reference

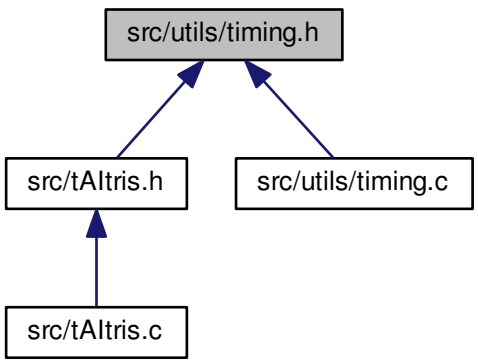
No description.

```
#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