

Gomoku AI

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 AIStats Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 cache_hits	5
3.1.2.2 nodes_searched	5
3.1.2.3 pruned	6
3.1.2.4 time_taken	6
3.2 Direction Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 dx	6
3.2.2.2 dy	6
3.3 GomokuGame Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Member Data Documentation	7
3.3.2.1 board	7
3.3.2.2 current_player	7
3.3.2.3 mode_ai	7
3.3.2.4 rule_captures	8
3.3.2.5 rule_center_opening	8
3.3.2.6 rule_no_double_threes	8
3.3.2.7 taken_stones	8
3.4 Move Struct Reference	8
3.4.1 Detailed Description	8
3.4.2 Member Data Documentation	8
3.4.2.1 col	9
3.4.2.2 row	9
3.4.2.3 score	9
3.5 TTEEntry Struct Reference	9
3.5.1 Detailed Description	10
3.5.2 Member Data Documentation	10
3.5.2.1 depth	10
3.5.2.2 key	10
3.5.2.3 move	10
3.5.2.4 score	10
3.6 VisualMarker Struct Reference	10

3.6.1 Detailed Description	11
3.6.2 Member Data Documentation	11
3.6.2.1 col	11
3.6.2.2 color	11
3.6.2.3 row	11
4 File Documentation	13
4.1 includes/ai.h File Reference	13
4.1.1 Function Documentation	14
4.1.1.1 ai_cleanup()	14
4.1.1.2 ai_evaluate_position()	14
4.1.1.3 ai_evaluate_position_for_player()	15
4.1.1.4 ai_generate_moves()	15
4.1.1.5 ai_get_best_move()	15
4.1.1.6 ai_get_last_stats()	16
4.1.1.7 ai_init()	16
4.2 includes/game.h File Reference	16
4.2.1 Function Documentation	17
4.2.1.1 game_capture_stones()	17
4.2.1.2 game_check_winner()	18
4.2.1.3 game_copy()	18
4.2.1.4 game_hash()	19
4.2.1.5 game_init()	19
4.2.1.6 game_is_double_free_three()	19
4.2.1.7 game_is_valid_position()	20
4.2.1.8 game_place_stone()	20
4.2.1.9 game_print_board()	21
4.3 includes/graphics.h File Reference	21
4.3.1 Function Documentation	22
4.3.1.1 graphics_add_visual_marker()	22
4.3.1.2 graphics_cleanup()	23
4.3.1.3 graphics_clear_visual_markers()	23
4.3.1.4 graphics_draw_game()	23
4.3.1.5 graphics_get_renderer()	23
4.3.1.6 graphics_get_window()	24
4.3.1.7 graphics_handle_click()	24
4.3.1.8 graphics_init()	24
4.3.1.9 graphics_remove_visual_marker()	25
4.3.1.10 graphics_show_winner()	26
4.3.2 Variable Documentation	26
4.3.2.1 button_rects	26
4.4 includes/types.h File Reference	26

4.4.1 Macro Definition Documentation	28
4.4.1.1 AI_INFINITY	28
4.4.1.2 BLACK	28
4.4.1.3 BOARD_SIZE	28
4.4.1.4 CELL_SIZE	28
4.4.1.5 EMPTY	28
4.4.1.6 M_PI	28
4.4.1.7 MAX_MOVES	29
4.4.1.8 MAX_TT_SIZE	29
4.4.1.9 MAX_VISUAL_MARKERS	29
4.4.1.10 PATTERN_FOUR	29
4.4.1.11 PATTERN_ONE	29
4.4.1.12 PATTERN_THREE	29
4.4.1.13 PATTERN_TWO	29
4.4.1.14 PATTERN_WIN	29
4.4.1.15 WHITE	30
4.4.1.16 WINDOW_HEIGHT	30
4.4.1.17 WINDOW_WIDTH	30
4.4.2 Variable Documentation	30
4.4.2.1 COLOR_BACKGROUND	30
4.4.2.2 COLOR_BLACK	30
4.4.2.3 COLOR_BLUE	30
4.4.2.4 COLOR_RED	30
4.4.2.5 COLOR_WHITE	31
4.5 src/ai.c File Reference	31
4.5.1 Function Documentation	32
4.5.1.1 ai_cleanup()	32
4.5.1.2 ai_evaluate_position()	32
4.5.1.3 ai_evaluate_position_for_player()	33
4.5.1.4 ai_generate_moves()	33
4.5.1.5 ai_get_best_move()	33
4.5.1.6 ai_get_last_stats()	34
4.5.1.7 ai_init()	34
4.5.1.8 count_consecutive_optimized()	34
4.5.1.9 evaluate_line_optimized()	35
4.5.1.10 find_blocking_moves_smart()	35
4.5.1.11 find_neighbor_positions_smart()	35
4.5.1.12 find_winning_moves_smart()	35
4.5.1.13 game_hash_optimized()	35
4.5.1.14 is_immediate_threat()	36
4.5.1.15 is_winning_move_fast()	36
4.5.1.16 minimax_balanced()	36

4.5.2 Variable Documentation	36
4.5.2.1 directions	36
4.5.2.2 last_stats	36
4.5.2.3 transposition_table	37
4.6 src/game.c File Reference	37
4.6.1 Function Documentation	38
4.6.1.1 game_capture_stones()	38
4.6.1.2 game_check_winner()	38
4.6.1.3 game_copy()	39
4.6.1.4 game_hash()	39
4.6.1.5 game_init()	39
4.6.1.6 game_is_double_free_three()	39
4.6.1.7 game_is_valid_position()	40
4.6.1.8 game_place_stone()	40
4.6.1.9 game_print_board()	41
4.6.2 Variable Documentation	41
4.6.2.1 directions	41
4.7 src/graphics.c File Reference	41
4.7.1 Macro Definition Documentation	43
4.7.1.1 BUTTON_HEIGHT	43
4.7.1.2 BUTTON_MARGIN	43
4.7.1.3 BUTTON_WIDTH	43
4.7.2 Function Documentation	43
4.7.2.1 draw_board()	44
4.7.2.2 draw_circle_filled()	44
4.7.2.3 draw_rule_buttons()	44
4.7.2.4 draw_simple_number()	44
4.7.2.5 draw_simple_text()	44
4.7.2.6 draw_stone()	44
4.7.2.7 draw_visual_markers()	45
4.7.2.8 graphics_add_visual_marker()	45
4.7.2.9 graphics_cleanup()	45
4.7.2.10 graphics_clear_visual_markers()	45
4.7.2.11 graphics_draw_game()	45
4.7.2.12 graphics_get_renderer()	46
4.7.2.13 graphics_get_window()	46
4.7.2.14 graphics_handle_click()	46
4.7.2.15 graphics_init()	47
4.7.2.16 graphics_remove_visual_marker()	47
4.7.2.17 graphics_show_winner()	47
4.7.3 Variable Documentation	47
4.7.3.1 button_rects	48

4.7.3.2 COLOR_BACKGROUND	48
4.7.3.3 COLOR_BLACK	48
4.7.3.4 COLOR_BLUE	48
4.7.3.5 COLOR_RED	48
4.7.3.6 COLOR_WHITE	48
4.7.3.7 font	48
4.7.3.8 renderer	49
4.7.3.9 visual_marker_count	49
4.7.3.10 visual_markers	49
4.7.3.11 window	49
4.8 src/main.c File Reference	49
4.8.1 Macro Definition Documentation	50
4.8.1.1 AI_PLAYER	50
4.8.1.2 AI_SEARCH_DEPTH	50
4.8.2 Function Documentation	50
4.8.2.1 get_rule_button_clicked()	50
4.8.2.2 handle_ai_move()	50
4.8.2.3 handle_player_move()	51
4.8.2.4 main()	51
4.8.2.5 print_game_info()	51
Index	53

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AIStats	AI statistics	5
Direction	Direction structure for pattern detection	6
GomokuGame	Game state structure	7
Move	Move structure	8
TTEntry	Transposition table entry	9
VisualMarker	Visual marker for AI visualization	10

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

includes/ ai.h	13
includes/ game.h	16
includes/ graphics.h	21
includes/ types.h	26
src/ ai.c	31
src/ game.c	37
src/ graphics.c	41
src/ main.c	49

Chapter 3

Class Documentation

3.1 AIStats Struct Reference

AI statistics.

```
#include <types.h>
```

Public Attributes

- int [nodes_searched](#)
- int [cache_hits](#)
- int [pruned](#)
- double [time_taken](#)

3.1.1 Detailed Description

AI statistics.

3.1.2 Member Data Documentation

3.1.2.1 [cache_hits](#)

```
int AIStats::cache_hits
```

3.1.2.2 [nodes_searched](#)

```
int AIStats::nodes_searched
```

3.1.2.3 pruned

```
int AStats::pruned
```

3.1.2.4 time_taken

```
double AStats::time_taken
```

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

- [includes/types.h](#)

3.2 Direction Struct Reference

[Direction](#) structure for pattern detection.

```
#include <types.h>
```

Public Attributes

- [int dx](#)
- [int dy](#)

3.2.1 Detailed Description

[Direction](#) structure for pattern detection.

3.2.2 Member Data Documentation

3.2.2.1 dx

```
int Direction::dx
```

3.2.2.2 dy

```
int Direction::dy
```

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

- [includes/types.h](#)

3.3 GomokuGame Struct Reference

Game state structure.

```
#include <types.h>
```

Public Attributes

- int [board](#) [BOARD_SIZE][BOARD_SIZE]
- int [current_player](#)
- int [taken_stones](#) [2]
- bool [rule_center_opening](#)
- bool [rule_no_double_threes](#)
- bool [rule_captures](#)
- bool [mode_ai](#)

3.3.1 Detailed Description

Game state structure.

3.3.2 Member Data Documentation

3.3.2.1 board

```
int GomokuGame::board[BOARD_SIZE][BOARD_SIZE]
```

3.3.2.2 current_player

```
int GomokuGame::current_player
```

3.3.2.3 mode_ai

```
bool GomokuGame::mode_ai
```

3.3.2.4 rule_captures

```
bool GomokuGame::rule_captures
```

3.3.2.5 rule_center_opening

```
bool GomokuGame::rule_center_opening
```

3.3.2.6 rule_no_double_threes

```
bool GomokuGame::rule_no_double_threes
```

3.3.2.7 taken_stones

```
int GomokuGame::taken_stones[2]
```

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

- [includes/types.h](#)

3.4 Move Struct Reference

[Move](#) structure.

```
#include <types.h>
```

Public Attributes

- int [row](#)
- int [col](#)
- int [score](#)

3.4.1 Detailed Description

[Move](#) structure.

3.4.2 Member Data Documentation

3.4.2.1 col

```
int Move::col
```

3.4.2.2 row

```
int Move::row
```

3.4.2.3 score

```
int Move::score
```

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

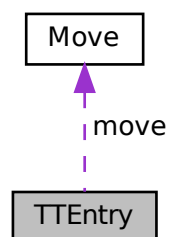
- [includes/types.h](#)

3.5 TTEntry Struct Reference

Transposition table entry.

```
#include <types.h>
```

Collaboration diagram for TTEntry:



Public Attributes

- [uint64_t key](#)
- [int depth](#)
- [int score](#)
- [Move move](#)

3.5.1 Detailed Description

Transposition table entry.

3.5.2 Member Data Documentation

3.5.2.1 depth

```
int TTEnter::depth
```

3.5.2.2 key

```
uint64_t TTEnter::key
```

3.5.2.3 move

```
Move TTEnter::move
```

3.5.2.4 score

```
int TTEnter::score
```

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

- [includes/types.h](#)

3.6 VisualMarker Struct Reference

Visual marker for AI visualization.

```
#include <types.h>
```

Public Attributes

- int [row](#)
- int [col](#)
- SDL_Color [color](#)

3.6.1 Detailed Description

Visual marker for AI visualization.

3.6.2 Member Data Documentation

3.6.2.1 col

```
int VisualMarker::col
```

3.6.2.2 color

```
SDL_Color VisualMarker::color
```

3.6.2.3 row

```
int VisualMarker::row
```

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

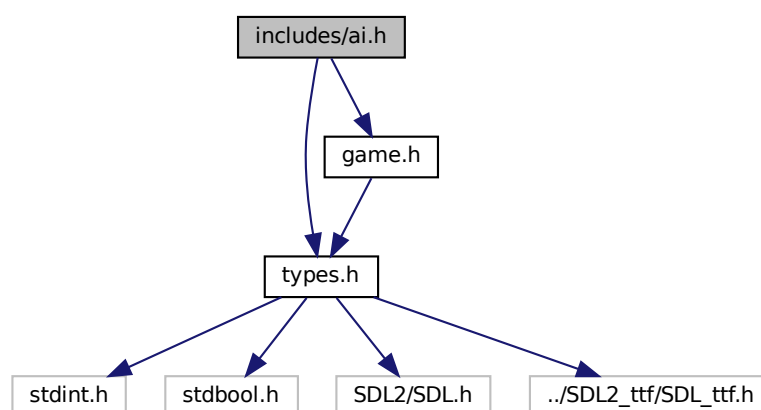
- [includes/types.h](#)

Chapter 4

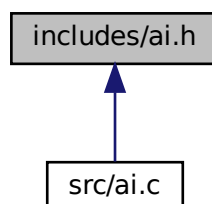
File Documentation

4.1 includes/ai.h File Reference

```
#include "types.h"  
#include "game.h"  
Include dependency graph for ai.h:
```



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



Functions

- void `ai_init` (void)
Initialize AI engine.
- void `ai_cleanup` (void)
Cleanup AI resources.
- `Move ai_get_best_move` (const `GomokuGame` *game, int depth, `AIStats` *stats)
Get best move for current player.
- int `ai_evaluate_position_for_player` (`GomokuGame` *game, int row, int col, int player)
Evaluate position for a specific player.
- void `ai_generate_moves` (const `GomokuGame` *game, `Move` *moves, int *move_count, int max_moves)
Generate candidate moves.
- int `ai_evaluate_position` (const `GomokuGame` *game)
Evaluate game position.
- `AIStats ai_get_last_stats` (void)
Get AI statistics from last search.

4.1.1 Function Documentation

4.1.1.1 `ai_cleanup()`

```
void ai_cleanup (
    void )
```

Cleanup AI resources.

4.1.1.2 `ai_evaluate_position()`

```
int ai_evaluate_position (
    const GomokuGame * game )
```

Evaluate game position.

Parameters

<i>game</i>	Game state
-------------	------------

Returns

Position evaluation score

4.1.1.3 ai_evaluate_position_for_player()

```
int ai_evaluate_position_for_player (
    GomokuGame * game,
    int row,
    int col,
    int player )
```

Evaluate position for a specific player.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position
<i>player</i>	Player to evaluate for

Returns

Position score

4.1.1.4 ai_generate_moves()

```
void ai_generate_moves (
    const GomokuGame * game,
    Move * moves,
    int * move_count,
    int max_moves )
```

Generate candidate moves.

Parameters

<i>game</i>	Game state
<i>moves</i>	Array to store generated moves
<i>move_count</i>	Pointer to store number of moves generated
<i>max_moves</i>	Maximum number of moves to generate

4.1.1.5 ai_get_best_move()

```
Move ai_get_best_move (
    const GomokuGame * game,
    int depth,
    AStats * stats )
```

Get best move for current player.

Parameters

<i>game</i>	Game state
<i>depth</i>	Search depth
<i>stats</i>	Pointer to store AI statistics (can be NULL)

Returns

Best move found

4.1.1.6 ai_get_last_stats()

```
AStats ai_get_last_stats (  
    void )
```

Get AI statistics from last search.

Returns

AI statistics

4.1.1.7 ai_init()

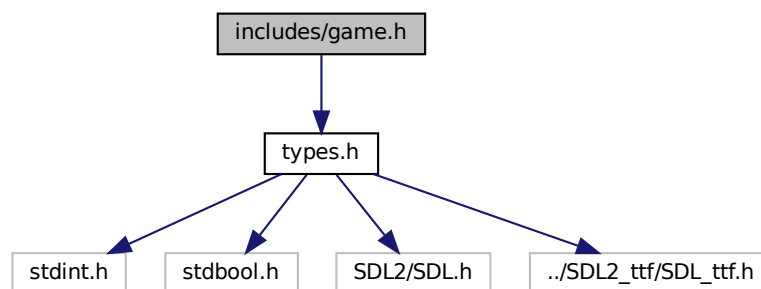
```
void ai_init (  
    void )
```

Initialize AI engine.

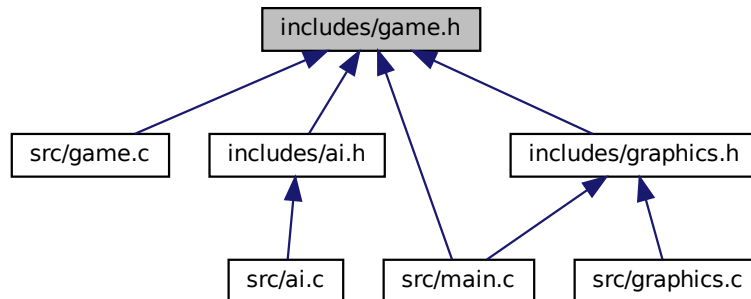
4.2 includes/game.h File Reference

```
#include "types.h"
```

Include dependency graph for game.h:



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



Functions

- void `game_init` (`GomokuGame` *game)
Initialize game state.
- void `game_copy` (`GomokuGame` *dest, const `GomokuGame` *src)
Copy game state.
- bool `game_place_stone` (`GomokuGame` *game, int row, int col)
Place a stone on the board.
- bool `game_check_winner` (const `GomokuGame` *game, int *winner)
Check if the game has a winner.
- bool `game_is_valid_position` (const `GomokuGame` *game, int row, int col)
Check if position is valid for placement.
- bool `game_is_double_free_three` (`GomokuGame` *game, int row, int col, int player)
Check if move creates double free three.
- int `game_capture_stones` (`GomokuGame` *game, int row, int col)
Capture stones around a placed stone.
- uint64_t `game_hash` (const `GomokuGame` *game)
Get hash value for game state.
- void `game_print_board` (const `GomokuGame` *game)
Print board state to console.

4.2.1 Function Documentation

4.2.1.1 `game_capture_stones()`

```

int game_capture_stones (
    GomokuGame * game,
    int row,
    int col )

```

Capture stones around a placed stone.

Parameters

<i>game</i>	Game state
<i>row</i>	Row of placed stone
<i>col</i>	Column of placed stone

Returns

Number of stones captured

4.2.1.2 game_check_winner()

```
bool game_check_winner (
    const GomokuGame * game,
    int * winner )
```

Check if the game has a winner.

Parameters

<i>game</i>	Game state
<i>winner</i>	Pointer to store winner (if any)

Returns

true if game is over, false otherwise

4.2.1.3 game_copy()

```
void game_copy (
    GomokuGame * dest,
    const GomokuGame * src )
```

Copy game state.

Parameters

<i>dest</i>	Destination game structure
<i>src</i>	Source game structure

4.2.1.4 game_hash()

```
uint64_t game_hash (
    const GomokuGame * game )
```

Get hash value for game state.

Parameters

<i>game</i>	Game state
-------------	------------

Returns

Hash value

4.2.1.5 game_init()

```
void game_init (
    GomokuGame * game )
```

Initialize game state.

Parameters

<i>game</i>	Pointer to game structure
-------------	---------------------------

4.2.1.6 game_is_double_free_three()

```
bool game_is_double_free_three (
    GomokuGame * game,
    int row,
    int col,
    int player )
```

Check if move creates double free three.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position
<i>player</i>	Player making the move

Returns

true if creates double free three, false otherwise

4.2.1.7 game_is_valid_position()

```
bool game_is_valid_position (
    const GomokuGame * game,
    int row,
    int col )
```

Check if position is valid for placement.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position

Returns

true if position is valid, false otherwise

4.2.1.8 game_place_stone()

```
bool game_place_stone (
    GomokuGame * game,
    int row,
    int col )
```

Place a stone on the board.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position

Returns

true if move was successful, false otherwise

4.2.1.9 game_print_board()

```
void game_print_board (
    const GomokuGame * game )
```

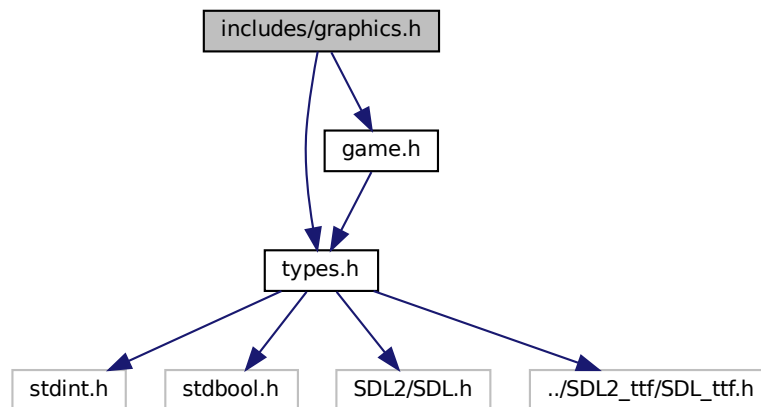
Print board state to console.

Parameters

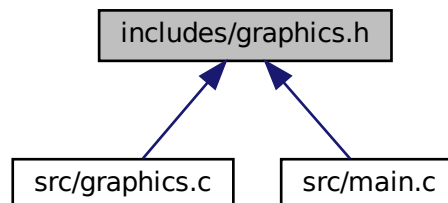
<i>game</i>	Game state
-------------	------------

4.3 includes/graphics.h File Reference

```
#include "types.h"
#include "game.h"
Include dependency graph for graphics.h:
```



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



Functions

- bool `graphics_init` (void)
Initialize graphics system.
- void `graphics_cleanup` (void)
Cleanup graphics resources.
- void `graphics_draw_game` (const GomokuGame *game)
Draw the complete game state.
- bool `graphics_handle_click` (int x, int y, int *row, int *col)
Handle mouse click events.
- void `graphics_add_visual_marker` (int row, int col, SDL_Color color)
Add visual marker for AI visualization.
- void `graphics_clear_visual_markers` (void)
Clear all visual markers.
- void `graphics_remove_visual_marker` (int row, int col)
Remove visual marker at specific position.
- void `graphics_show_winner` (int winner)
Show winner message.
- SDL_Renderer * `graphics_get_renderer` (void)
Get SDL renderer (for custom drawing)
- SDL_Window * `graphics_get_window` (void)
Get SDL window (for event handling)

Variables

- SDL_Rect `button_rects` [4]

4.3.1 Function Documentation

4.3.1.1 `graphics_add_visual_marker()`

```
void graphics_add_visual_marker (
    int row,
    int col,
    SDL_Color color )
```

Add visual marker for AI visualization.

Parameters

<i>row</i>	Board row
<i>col</i>	Board column
<i>color</i>	Marker color

4.3.1.2 graphics_cleanup()

```
void graphics_cleanup (
    void )
```

Cleanup graphics resources.

4.3.1.3 graphics_clear_visual_markers()

```
void graphics_clear_visual_markers (
    void )
```

Clear all visual markers.

4.3.1.4 graphics_draw_game()

```
void graphics_draw_game (
    const GomokuGame * game )
```

Draw the complete game state.

Parameters

<i>game</i>	Game state to draw
-------------	--------------------

4.3.1.5 graphics_get_renderer()

```
SDL_Renderer* graphics_get_renderer (
    void )
```

Get SDL renderer (for custom drawing)

Returns

SDL renderer pointer

4.3.1.6 graphics_get_window()

```
SDL_Window* graphics_get_window (
    void )
```

Get SDL window (for event handling)

Returns

SDL window pointer

4.3.1.7 graphics_handle_click()

```
bool graphics_handle_click (
    int x,
    int y,
    int * row,
    int * col )
```

Handle mouse click events.

Parameters

<i>x</i>	Mouse x coordinate
<i>y</i>	Mouse y coordinate
<i>row</i>	Pointer to store board row
<i>col</i>	Pointer to store board column

Returns

true if click is on valid board position

4.3.1.8 graphics_init()

```
bool graphics_init (
    void )
```

Initialize graphics system.

Returns

true if successful, false otherwise

4.3.1.9 graphics_remove_visual_marker()

```
void graphics_remove_visual_marker (
    int row,
    int col )
```

Remove visual marker at specific position.

Parameters

<i>row</i>	Board row
<i>col</i>	Board column

4.3.1.10 graphics_show_winner()

```
void graphics_show_winner (
    int winner )
```

Show winner message.

Parameters

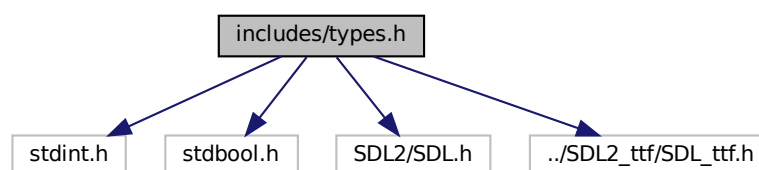
<i>winner</i>	Winner player (BLACK, WHITE, or EMPTY for draw)
---------------	---

4.3.2 Variable Documentation**4.3.2.1 button_rects**

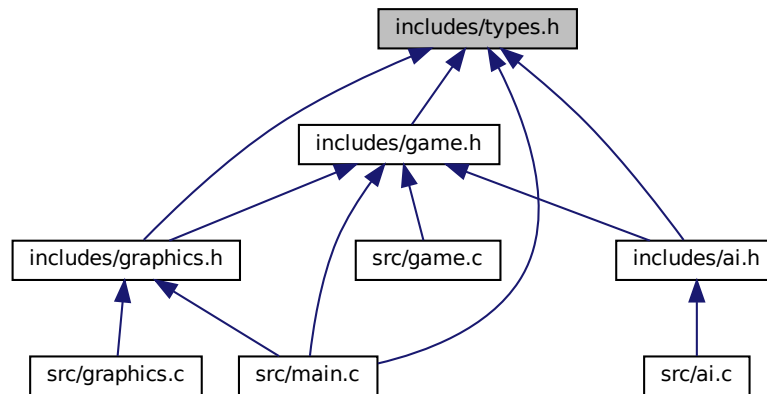
```
SDL_Rect button_rects[4] [extern]
```

4.4 includes/types.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <SDL2/SDL.h>
#include "../SDL2_ttf/SDL_ttf.h"
Include dependency graph for types.h:
```



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



Classes

- struct [Direction](#)
Direction structure for pattern detection.
- struct [Move](#)
Move structure.
- struct [GomokuGame](#)
Game state structure.
- struct [TTEEntry](#)
Transposition table entry.
- struct [VisualMarker](#)
Visual marker for AI visualization.
- struct [AIStats](#)
AI statistics.

Macros

- #define [BOARD_SIZE](#) 19
- #define [CELL_SIZE](#) 40
- #define [WINDOW_WIDTH](#) 1000
- #define [WINDOW_HEIGHT](#) 800
- #define [AI_INFINITY](#) 1000000000
- #define [MAX_TT_SIZE](#) 500000
- #define [MAX_MOVES](#) 15
- #define [MAX_VISUAL_MARKERS](#) 1000
- #define [PATTERN_WIN](#) 100000
- #define [PATTERN_FOUR](#) 50000
- #define [PATTERN_THREE](#) 8000
- #define [PATTERN_TWO](#) 800
- #define [PATTERN_ONE](#) 80
- #define [EMPTY](#) 0
- #define [BLACK](#) 1
- #define [WHITE](#) 2
- #define [M_PI](#) 3.14159265358979323846

Variables

- const SDL_Color [COLOR_BLACK](#)
- const SDL_Color [COLOR_WHITE](#)
- const SDL_Color [COLOR_BLUE](#)
- const SDL_Color [COLOR_RED](#)
- const SDL_Color [COLOR_BACKGROUND](#)

4.4.1 Macro Definition Documentation

4.4.1.1 AI_INFINITY

```
#define AI_INFINITY 1000000000
```

4.4.1.2 BLACK

```
#define BLACK 1
```

4.4.1.3 BOARD_SIZE

```
#define BOARD_SIZE 19
```

4.4.1.4 CELL_SIZE

```
#define CELL_SIZE 40
```

4.4.1.5 EMPTY

```
#define EMPTY 0
```

4.4.1.6 M_PI

```
#define M_PI 3.14159265358979323846
```

4.4.1.7 MAX_MOVES

```
#define MAX_MOVES 15
```

4.4.1.8 MAX_TT_SIZE

```
#define MAX_TT_SIZE 500000
```

4.4.1.9 MAX_VISUAL_MARKERS

```
#define MAX_VISUAL_MARKERS 1000
```

4.4.1.10 PATTERN_FOUR

```
#define PATTERN_FOUR 50000
```

4.4.1.11 PATTERN_ONE

```
#define PATTERN_ONE 80
```

4.4.1.12 PATTERN_THREE

```
#define PATTERN_THREE 8000
```

4.4.1.13 PATTERN_TWO

```
#define PATTERN_TWO 800
```

4.4.1.14 PATTERN_WIN

```
#define PATTERN_WIN 100000
```

4.4.1.15 WHITE

```
#define WHITE 2
```

4.4.1.16 WINDOW_HEIGHT

```
#define WINDOW_HEIGHT 800
```

4.4.1.17 WINDOW_WIDTH

```
#define WINDOW_WIDTH 1000
```

4.4.2 Variable Documentation

4.4.2.1 COLOR_BACKGROUND

```
const SDL_Color COLOR_BACKGROUND [extern]
```

4.4.2.2 COLOR_BLACK

```
const SDL_Color COLOR_BLACK [extern]
```

4.4.2.3 COLOR_BLUE

```
const SDL_Color COLOR_BLUE [extern]
```

4.4.2.4 COLOR_RED

```
const SDL_Color COLOR_RED [extern]
```

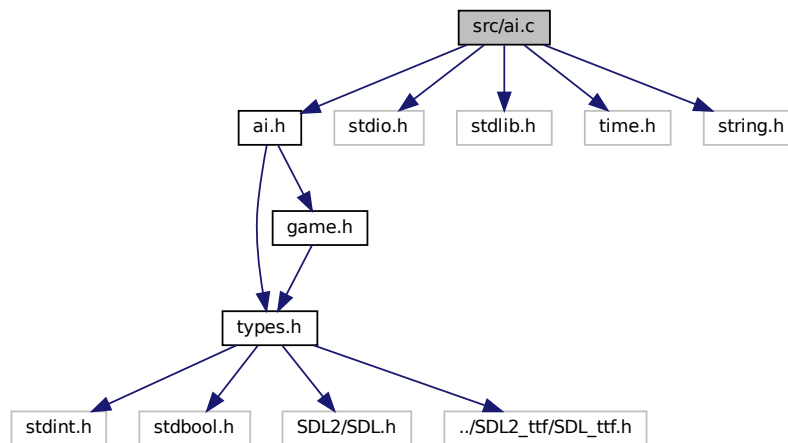
4.4.2.5 COLOR_WHITE

```
const SDL_Color COLOR_WHITE [extern]
```

4.5 src/ai.c File Reference

```
#include "ai.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
```

Include dependency graph for ai.c:



Functions

- static int [minimax_balanced](#) (GomokuGame *game, int depth, int alpha, int beta, bool maximizing, [Move](#) *best_move)
- static int [count_consecutive_optimized](#) (const GomokuGame *game, int row, int col, int dx, int dy, int player)
- static int [evaluate_line_optimized](#) (const GomokuGame *game, int row, int col, int dx, int dy, int player)
- static void [find_winning_moves_smart](#) (const GomokuGame *game, [Move](#) *moves, int *count, int player)
- static void [find_blocking_moves_smart](#) (const GomokuGame *game, [Move](#) *moves, int *count, int player)
- static void [find_neighbor_positions_smart](#) (const GomokuGame *game, [Move](#) *moves, int *count, int max↔_moves)
- static uint64_t [game_hash_optimized](#) (const GomokuGame *game)
- static bool [is_immediate_threat](#) (const GomokuGame *game, int row, int col, int player)
- void [ai_init](#) (void)
Initialize AI engine.
- void [ai_cleanup](#) (void)
Cleanup AI resources.
- int [ai_evaluate_position_for_player](#) (GomokuGame *game, int row, int col, int player)
Evaluate position for a specific player.
- static bool [is_winning_move_fast](#) (const GomokuGame *game, int row, int col, int player)

- void `ai_generate_moves` (const `GomokuGame` *game, `Move` *moves, int *move_count, int max_moves)
Generate candidate moves.
- int `ai_evaluate_position` (const `GomokuGame` *game)
Evaluate game position.
- `Move` `ai_get_best_move` (const `GomokuGame` *game, int depth, `AIStats` *stats)
Get best move for current player.
- `AIStats` `ai_get_last_stats` (void)
Get AI statistics from last search.

Variables

- static `TTEEntry` * `transposition_table` = NULL
- static `AIStats` `last_stats` = {0}
- static const `Direction` `directions` [4] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}}

4.5.1 Function Documentation

4.5.1.1 ai_cleanup()

```
void ai_cleanup (
    void )
```

Cleanup AI resources.

4.5.1.2 ai_evaluate_position()

```
int ai_evaluate_position (
    const GomokuGame * game )
```

Evaluate game position.

Parameters

<i>game</i>	Game state
-------------	------------

Returns

Position evaluation score

4.5.1.3 ai_evaluate_position_for_player()

```
int ai_evaluate_position_for_player (
    GomokuGame * game,
    int row,
    int col,
    int player )
```

Evaluate position for a specific player.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position
<i>player</i>	Player to evaluate for

Returns

Position score

4.5.1.4 ai_generate_moves()

```
void ai_generate_moves (
    const GomokuGame * game,
    Move * moves,
    int * move_count,
    int max_moves )
```

Generate candidate moves.

Parameters

<i>game</i>	Game state
<i>moves</i>	Array to store generated moves
<i>move_count</i>	Pointer to store number of moves generated
<i>max_moves</i>	Maximum number of moves to generate

4.5.1.5 ai_get_best_move()

```
Move ai_get_best_move (
    const GomokuGame * game,
    int depth,
    AStats * stats )
```

Get best move for current player.

Parameters

<i>game</i>	Game state
<i>depth</i>	Search depth
<i>stats</i>	Pointer to store AI statistics (can be NULL)

Returns

Best move found

4.5.1.6 ai_get_last_stats()

```
AIStruct ai_get_last_stats (  
    void )
```

Get AI statistics from last search.

Returns

AI statistics

4.5.1.7 ai_init()

```
void ai_init (  
    void )
```

Initialize AI engine.

4.5.1.8 count_consecutive_optimized()

```
static int count_consecutive_optimized (  
    const GomokuGame * game,  
    int row,  
    int col,  
    int dx,  
    int dy,  
    int player ) [inline], [static]
```

4.5.1.9 evaluate_line_optimized()

```
static int evaluate_line_optimized (
    const GomokuGame * game,
    int row,
    int col,
    int dx,
    int dy,
    int player ) [inline], [static]
```

4.5.1.10 find_blocking_moves_smart()

```
static void find_blocking_moves_smart (
    const GomokuGame * game,
    Move * moves,
    int * count,
    int player ) [static]
```

4.5.1.11 find_neighbor_positions_smart()

```
static void find_neighbor_positions_smart (
    const GomokuGame * game,
    Move * moves,
    int * count,
    int max_moves ) [static]
```

4.5.1.12 find_winning_moves_smart()

```
static void find_winning_moves_smart (
    const GomokuGame * game,
    Move * moves,
    int * count,
    int player ) [static]
```

4.5.1.13 game_hash_optimized()

```
static uint64_t game_hash_optimized (
    const GomokuGame * game ) [static]
```

4.5.1.14 is_immediate_threat()

```
static bool is_immediate_threat (
    const GomokuGame * game,
    int row,
    int col,
    int player ) [static]
```

4.5.1.15 is_winning_move_fast()

```
static bool is_winning_move_fast (
    const GomokuGame * game,
    int row,
    int col,
    int player ) [inline], [static]
```

4.5.1.16 minimax_balanced()

```
static int minimax_balanced (
    GomokuGame * game,
    int depth,
    int alpha,
    int beta,
    bool maximizing,
    Move * best_move ) [static]
```

4.5.2 Variable Documentation

4.5.2.1 directions

```
const Direction directions[4] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}} [static]
```

4.5.2.2 last_stats

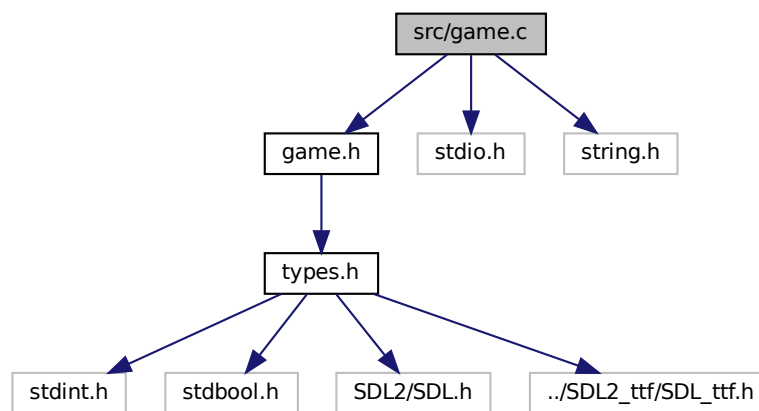
```
AISStats last_stats = {0} [static]
```

4.5.2.3 transposition_table

```
TTEntry* transposition_table = NULL [static]
```

4.6 src/game.c File Reference

```
#include "game.h"
#include <stdio.h>
#include <string.h>
Include dependency graph for game.c:
```



Functions

- void `game_init` (`GomokuGame *game`)
Initialize game state.
- void `game_copy` (`GomokuGame *dest`, const `GomokuGame *src`)
Copy game state.
- bool `game_is_valid_position` (const `GomokuGame *game`, int row, int col)
Check if position is valid for placement.
- bool `game_is_double_free_three` (`GomokuGame *game`, int row, int col, int player)
Check if move creates double free three.
- int `game_capture_stones` (`GomokuGame *game`, int row, int col)
Capture stones around a placed stone.
- bool `game_place_stone` (`GomokuGame *game`, int row, int col)
Place a stone on the board.
- bool `game_check_winner` (const `GomokuGame *game`, int *winner)
Check if the game has a winner.
- uint64_t `game_hash` (const `GomokuGame *game`)
Get hash value for game state.
- void `game_print_board` (const `GomokuGame *game`)
Print board state to console.

Variables

- static const [Direction](#) [directions](#) [4] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}}

4.6.1 Function Documentation

4.6.1.1 `game_capture_stones()`

```
int game_capture_stones (
    GomokuGame * game,
    int row,
    int col )
```

Capture stones around a placed stone.

Parameters

<i>game</i>	Game state
<i>row</i>	Row of placed stone
<i>col</i>	Column of placed stone

Returns

Number of stones captured

4.6.1.2 `game_check_winner()`

```
bool game_check_winner (
    const GomokuGame * game,
    int * winner )
```

Check if the game has a winner.

Parameters

<i>game</i>	Game state
<i>winner</i>	Pointer to store winner (if any)

Returns

true if game is over, false otherwise

4.6.1.3 game_copy()

```
void game_copy (
    GomokuGame * dest,
    const GomokuGame * src )
```

Copy game state.

Parameters

<i>dest</i>	Destination game structure
<i>src</i>	Source game structure

4.6.1.4 game_hash()

```
uint64_t game_hash (
    const GomokuGame * game )
```

Get hash value for game state.

Parameters

<i>game</i>	Game state
-------------	------------

Returns

Hash value

4.6.1.5 game_init()

```
void game_init (
    GomokuGame * game )
```

Initialize game state.

Parameters

<i>game</i>	Pointer to game structure
-------------	---------------------------

4.6.1.6 game_is_double_free_three()

```
bool game_is_double_free_three (
```

```
GomokuGame * game,  
int row,  
int col,  
int player )
```

Check if move creates double free three.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position
<i>player</i>	Player making the move

Returns

true if creates double free three, false otherwise

4.6.1.7 game_is_valid_position()

```
bool game_is_valid_position (  
    const GomokuGame * game,  
    int row,  
    int col )
```

Check if position is valid for placement.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position

Returns

true if position is valid, false otherwise

4.6.1.8 game_place_stone()

```
bool game_place_stone (  
    GomokuGame * game,  
    int row,  
    int col )
```

Place a stone on the board.

Parameters

<i>game</i>	Game state
<i>row</i>	Row position
<i>col</i>	Column position

Returns

true if move was successful, false otherwise

4.6.1.9 game_print_board()

```
void game_print_board (
    const GomokuGame * game )
```

Print board state to console.

Parameters

<i>game</i>	Game state
-------------	------------

4.6.2 Variable Documentation**4.6.2.1 directions**

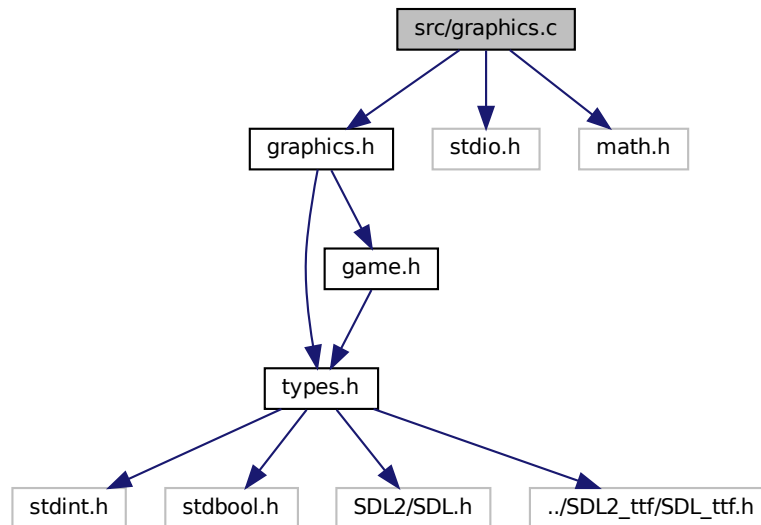
```
const Direction directions[4] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}} [static]
```

4.7 src/graphics.c File Reference

```
#include "graphics.h"
#include <stdio.h>
```

```
#include <math.h>
```

Include dependency graph for graphics.c:



Macros

- `#define` `BUTTON_WIDTH` 180
- `#define` `BUTTON_HEIGHT` 40
- `#define` `BUTTON_MARGIN` 10

Functions

- static void `draw_board` (void)
- static void `draw_stone` (int row, int col, `SDL_Color` color)
- static void `draw_circle_filled` (int x, int y, int radius, `SDL_Color` color)
- static void `draw_visual_markers` (void)
- static void `draw_simple_number` (int x, int y, int num)
- static void `draw_simple_text` (int x, int y, const char *text)
- static void `draw_rule_buttons` (const `GomokuGame` *game)
- bool `graphics_init` (void)
Initialize graphics system.
- void `graphics_cleanup` (void)
Cleanup graphics resources.
- void `graphics_draw_game` (const `GomokuGame` *game)
Draw the complete game state.
- bool `graphics_handle_click` (int x, int y, int *row, int *col)
Handle mouse click events.
- void `graphics_add_visual_marker` (int row, int col, `SDL_Color` color)
Add visual marker for AI visualization.
- void `graphics_clear_visual_markers` (void)

Clear all visual markers.

- void [graphics_remove_visual_marker](#) (int row, int col)

Remove visual marker at specific position.

- void [graphics_show_winner](#) (int winner)

Show winner message.

- SDL_Renderer * [graphics_get_renderer](#) (void)

Get SDL renderer (for custom drawing)

- SDL_Window * [graphics_get_window](#) (void)

Get SDL window (for event handling)

Variables

- const SDL_Color [COLOR_BLACK](#) = {0, 0, 0, 255}
- const SDL_Color [COLOR_WHITE](#) = {255, 255, 255, 255}
- const SDL_Color [COLOR_BLUE](#) = {0, 0, 255, 255}
- const SDL_Color [COLOR_RED](#) = {255, 0, 0, 255}
- const SDL_Color [COLOR_BACKGROUND](#) = {255, 190, 90, 255}
- static SDL_Window * [window](#) = NULL
- static SDL_Renderer * [renderer](#) = NULL
- static [VisualMarker](#) [visual_markers](#) [[MAX_VISUAL_MARKERS](#)]
- static int [visual_marker_count](#) = 0
- static TTF_Font * [font](#) = NULL
- SDL_Rect [button_rects](#) [4]

4.7.1 Macro Definition Documentation

4.7.1.1 BUTTON_HEIGHT

```
#define BUTTON_HEIGHT 40
```

4.7.1.2 BUTTON_MARGIN

```
#define BUTTON_MARGIN 10
```

4.7.1.3 BUTTON_WIDTH

```
#define BUTTON_WIDTH 180
```

4.7.2 Function Documentation

4.7.2.1 draw_board()

```
static void draw_board (
    void ) [static]
```

4.7.2.2 draw_circle_filled()

```
static void draw_circle_filled (
    int x,
    int y,
    int radius,
    SDL_Color color ) [static]
```

4.7.2.3 draw_rule_buttons()

```
void draw_rule_buttons (
    const GomokuGame * game ) [static]
```

4.7.2.4 draw_simple_number()

```
static void draw_simple_number (
    int x,
    int y,
    int num ) [static]
```

4.7.2.5 draw_simple_text()

```
static void draw_simple_text (
    int x,
    int y,
    const char * text ) [static]
```

4.7.2.6 draw_stone()

```
static void draw_stone (
    int row,
    int col,
    SDL_Color color ) [static]
```

4.7.2.7 draw_visual_markers()

```
static void draw_visual_markers (
    void ) [static]
```

4.7.2.8 graphics_add_visual_marker()

```
void graphics_add_visual_marker (
    int row,
    int col,
    SDL_Color color )
```

Add visual marker for AI visualization.

Parameters

<i>row</i>	Board row
<i>col</i>	Board column
<i>color</i>	Marker color

4.7.2.9 graphics_cleanup()

```
void graphics_cleanup (
    void )
```

Cleanup graphics resources.

4.7.2.10 graphics_clear_visual_markers()

```
void graphics_clear_visual_markers (
    void )
```

Clear all visual markers.

4.7.2.11 graphics_draw_game()

```
void graphics_draw_game (
    const GomokuGame * game )
```

Draw the complete game state.

Parameters

<i>game</i>	Game state to draw
-------------	--------------------

4.7.2.12 graphics_get_renderer()

```
SDL_Renderer* graphics_get_renderer (
    void )
```

Get SDL renderer (for custom drawing)

Returns

SDL renderer pointer

4.7.2.13 graphics_get_window()

```
SDL_Window* graphics_get_window (
    void )
```

Get SDL window (for event handling)

Returns

SDL window pointer

4.7.2.14 graphics_handle_click()

```
bool graphics_handle_click (
    int x,
    int y,
    int * row,
    int * col )
```

Handle mouse click events.

Parameters

<i>x</i>	Mouse x coordinate
<i>y</i>	Mouse y coordinate
<i>row</i>	Pointer to store board row
<i>col</i>	Pointer to store board column

Returns

true if click is on valid board position

4.7.2.15 graphics_init()

```
bool graphics_init (
    void )
```

Initialize graphics system.

Returns

true if successful, false otherwise

4.7.2.16 graphics_remove_visual_marker()

```
void graphics_remove_visual_marker (
    int row,
    int col )
```

Remove visual marker at specific position.

Parameters

<i>row</i>	Board row
<i>col</i>	Board column

4.7.2.17 graphics_show_winner()

```
void graphics_show_winner (
    int winner )
```

Show winner message.

Parameters

<i>winner</i>	Winner player (BLACK, WHITE, or EMPTY for draw)
---------------	---

4.7.3 Variable Documentation

4.7.3.1 button_rects

```
SDL_Rect button_rects[4]
```

Initial value:

```
= {  
    {CELL_SIZE * BOARD_SIZE + 20, 150, 180, 40},  
    {CELL_SIZE * BOARD_SIZE + 20, 200, 180, 40},  
    {CELL_SIZE * BOARD_SIZE + 20, 250, 180, 40},  
    {CELL_SIZE * BOARD_SIZE + 20, 300, 180, 40}  
}
```

4.7.3.2 COLOR_BACKGROUND

```
const SDL_Color COLOR_BACKGROUND = {255, 190, 90, 255}
```

4.7.3.3 COLOR_BLACK

```
const SDL_Color COLOR_BLACK = {0, 0, 0, 255}
```

4.7.3.4 COLOR_BLUE

```
const SDL_Color COLOR_BLUE = {0, 0, 255, 255}
```

4.7.3.5 COLOR_RED

```
const SDL_Color COLOR_RED = {255, 0, 0, 255}
```

4.7.3.6 COLOR_WHITE

```
const SDL_Color COLOR_WHITE = {255, 255, 255, 255}
```

4.7.3.7 font

```
TTF_Font* font = NULL [static]
```


4.7.3.8 renderer

```
SDL_Renderer* renderer = NULL [static]
```

4.7.3.9 visual_marker_count

```
int visual_marker_count = 0 [static]
```

4.7.3.10 visual_markers

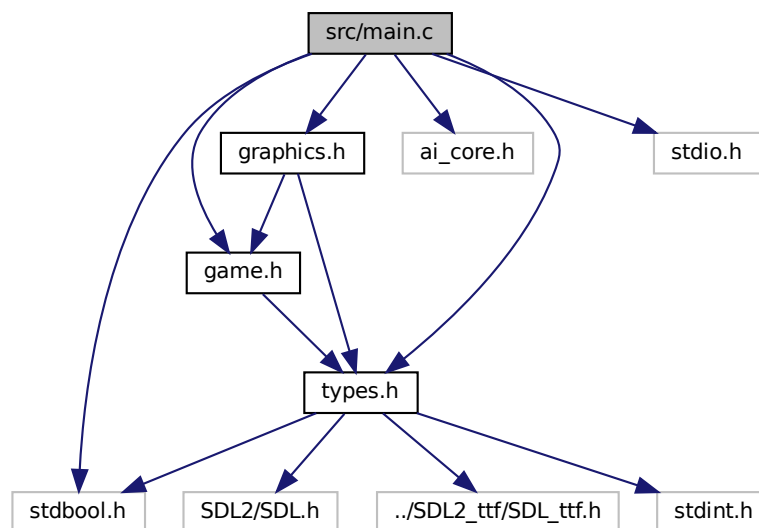
```
VisualMarker visual_markers[MAX_VISUAL_MARKERS] [static]
```

4.7.3.11 window

```
SDL_Window* window = NULL [static]
```

4.8 src/main.c File Reference

```
#include "types.h"  
#include "game.h"  
#include "ai_core.h"  
#include "graphics.h"  
#include <stdio.h>  
#include <stdbool.h>  
Include dependency graph for main.c:
```



Macros

- `#define AI_SEARCH_DEPTH 10`
- `#define AI_PLAYER WHITE`

Functions

- static void `handle_player_move` (`GomokuGame` *game, int row, int col)
- static void `handle_ai_move` (`GomokuGame` *game)
- static void `print_game_info` (const `GomokuGame` *game)
- static int `get_rule_button_clicked` (int x, int y)
- int `main` (void)

4.8.1 Macro Definition Documentation

4.8.1.1 AI_PLAYER

```
#define AI_PLAYER WHITE
```

4.8.1.2 AI_SEARCH_DEPTH

```
#define AI_SEARCH_DEPTH 10
```

4.8.2 Function Documentation

4.8.2.1 get_rule_button_clicked()

```
static int get_rule_button_clicked (  
    int x,  
    int y ) [static]
```

4.8.2.2 handle_ai_move()

```
static void handle_ai_move (  
    GomokuGame * game ) [static]
```

4.8.2.3 handle_player_move()

```
static void handle_player_move (  
    GomokuGame * game,  
    int row,  
    int col ) [static]
```

4.8.2.4 main()

```
int main (  
    void )
```

4.8.2.5 print_game_info()

```
static void print_game_info (  
    const GomokuGame * game ) [static]
```


Index

ai.c

- ai_cleanup, 32
- ai_evaluate_position, 32
- ai_evaluate_position_for_player, 32
- ai_generate_moves, 33
- ai_get_best_move, 33
- ai_get_last_stats, 34
- ai_init, 34
- count_consecutive_optimized, 34
- directions, 36
- evaluate_line_optimized, 34
- find_blocking_moves_smart, 35
- find_neighbor_positions_smart, 35
- find_winning_moves_smart, 35
- game_hash_optimized, 35
- is_immediate_threat, 35
- is_winning_move_fast, 36
- last_stats, 36
- minimax_balanced, 36
- transposition_table, 36

ai.h

- ai_cleanup, 14
- ai_evaluate_position, 14
- ai_evaluate_position_for_player, 14
- ai_generate_moves, 15
- ai_get_best_move, 15
- ai_get_last_stats, 16
- ai_init, 16

ai_cleanup

- ai.c, 32
- ai.h, 14

ai_evaluate_position

- ai.c, 32
- ai.h, 14

ai_evaluate_position_for_player

- ai.c, 32
- ai.h, 14

ai_generate_moves

- ai.c, 33
- ai.h, 15

ai_get_best_move

- ai.c, 33
- ai.h, 15

ai_get_last_stats

- ai.c, 34
- ai.h, 16

AI_INFINITY

- types.h, 28

ai_init

- ai.c, 34

- ai.h, 16

AI_PLAYER

- main.c, 50

AI_SEARCH_DEPTH

- main.c, 50

AIStats, 5

- cache_hits, 5
- nodes_searched, 5
- pruned, 5
- time_taken, 6

BLACK

- types.h, 28

board

- GomokuGame, 7

BOARD_SIZE

- types.h, 28

BUTTON_HEIGHT

- graphics.c, 43

BUTTON_MARGIN

- graphics.c, 43

button_rects

- graphics.c, 47
- graphics.h, 26

BUTTON_WIDTH

- graphics.c, 43

cache_hits

- AIStats, 5

CELL_SIZE

- types.h, 28

col

- Move, 8
- VisualMarker, 11

color

- VisualMarker, 11

COLOR_BACKGROUND

- graphics.c, 48
- types.h, 30

COLOR_BLACK

- graphics.c, 48
- types.h, 30

COLOR_BLUE

- graphics.c, 48
- types.h, 30

COLOR_RED

- graphics.c, 48
- types.h, 30

COLOR_WHITE

- graphics.c, 48
- types.h, 30
- count_consecutive_optimized
 - ai.c, 34
- current_player
 - GomokuGame, 7
- depth
 - TTEnter, 10
- Direction, 6
 - dx, 6
 - dy, 6
- directions
 - ai.c, 36
 - game.c, 41
- draw_board
 - graphics.c, 43
- draw_circle_filled
 - graphics.c, 44
- draw_rule_buttons
 - graphics.c, 44
- draw_simple_number
 - graphics.c, 44
- draw_simple_text
 - graphics.c, 44
- draw_stone
 - graphics.c, 44
- draw_visual_markers
 - graphics.c, 44
- dx
 - Direction, 6
- dy
 - Direction, 6
- EMPTY
 - types.h, 28
- evaluate_line_optimized
 - ai.c, 34
- find_blocking_moves_smart
 - ai.c, 35
- find_neighbor_positions_smart
 - ai.c, 35
- find_winning_moves_smart
 - ai.c, 35
- font
 - graphics.c, 48
- game.c
 - directions, 41
 - game_capture_stones, 38
 - game_check_winner, 38
 - game_copy, 38
 - game_hash, 39
 - game_init, 39
 - game_is_double_free_three, 39
 - game_is_valid_position, 40
 - game_place_stone, 40
 - game_print_board, 41
- game.h
 - game_capture_stones, 17
 - game_check_winner, 18
 - game_copy, 18
 - game_hash, 18
 - game_init, 19
 - game_is_double_free_three, 19
 - game_is_valid_position, 20
 - game_place_stone, 20
 - game_print_board, 20
- game_capture_stones
 - game.c, 38
 - game.h, 17
- game_check_winner
 - game.c, 38
 - game.h, 18
- game_copy
 - game.c, 38
 - game.h, 18
- game_hash
 - game.c, 39
 - game.h, 18
- game_hash_optimized
 - ai.c, 35
- game_init
 - game.c, 39
 - game.h, 19
- game_is_double_free_three
 - game.c, 39
 - game.h, 19
- game_is_valid_position
 - game.c, 40
 - game.h, 20
- game_place_stone
 - game.c, 40
 - game.h, 20
- game_print_board
 - game.c, 41
 - game.h, 20
- get_rule_button_clicked
 - main.c, 50
- GomokuGame, 7
 - board, 7
 - current_player, 7
 - mode_ai, 7
 - rule_captures, 7
 - rule_center_opening, 8
 - rule_no_double_threes, 8
 - taken_stones, 8
- graphics.c
 - BUTTON_HEIGHT, 43
 - BUTTON_MARGIN, 43
 - button_rects, 47
 - BUTTON_WIDTH, 43
 - COLOR_BACKGROUND, 48
 - COLOR_BLACK, 48
 - COLOR_BLUE, 48
 - COLOR_RED, 48

- COLOR_WHITE, 48
- draw_board, 43
- draw_circle_filled, 44
- draw_rule_buttons, 44
- draw_simple_number, 44
- draw_simple_text, 44
- draw_stone, 44
- draw_visual_markers, 44
- font, 48
- graphics_add_visual_marker, 45
- graphics_cleanup, 45
- graphics_clear_visual_markers, 45
- graphics_draw_game, 45
- graphics_get_renderer, 46
- graphics_get_window, 46
- graphics_handle_click, 46
- graphics_init, 47
- graphics_remove_visual_marker, 47
- graphics_show_winner, 47
- renderer, 48
- visual_marker_count, 49
- visual_markers, 49
- window, 49
- graphics.h
 - button_rects, 26
 - graphics_add_visual_marker, 22
 - graphics_cleanup, 22
 - graphics_clear_visual_markers, 23
 - graphics_draw_game, 23
 - graphics_get_renderer, 23
 - graphics_get_window, 23
 - graphics_handle_click, 24
 - graphics_init, 24
 - graphics_remove_visual_marker, 24
 - graphics_show_winner, 26
- graphics_add_visual_marker
 - graphics.c, 45
 - graphics.h, 22
- graphics_cleanup
 - graphics.c, 45
 - graphics.h, 22
- graphics_clear_visual_markers
 - graphics.c, 45
 - graphics.h, 23
- graphics_draw_game
 - graphics.c, 45
 - graphics.h, 23
- graphics_get_renderer
 - graphics.c, 46
 - graphics.h, 23
- graphics_get_window
 - graphics.c, 46
 - graphics.h, 23
- graphics_handle_click
 - graphics.c, 46
 - graphics.h, 24
- graphics_init
 - graphics.c, 47
 - graphics.h, 24
- graphics_remove_visual_marker
 - graphics.c, 47
 - graphics.h, 24
- graphics_show_winner
 - graphics.c, 47
 - graphics.h, 26
- handle_ai_move
 - main.c, 50
- handle_player_move
 - main.c, 50
- includes/ai.h, 13
- includes/game.h, 16
- includes/graphics.h, 21
- includes/types.h, 26
- is_immediate_threat
 - ai.c, 35
- is_winning_move_fast
 - ai.c, 36
- key
 - TTEEntry, 10
- last_stats
 - ai.c, 36
- M_PI
 - types.h, 28
- main
 - main.c, 51
- main.c
 - AI_PLAYER, 50
 - AI_SEARCH_DEPTH, 50
 - get_rule_button_clicked, 50
 - handle_ai_move, 50
 - handle_player_move, 50
 - main, 51
 - print_game_info, 51
- MAX_MOVES
 - types.h, 28
- MAX_TT_SIZE
 - types.h, 29
- MAX_VISUAL_MARKERS
 - types.h, 29
- minimax_balanced
 - ai.c, 36
- mode_ai
 - GomokuGame, 7
- Move, 8
 - col, 8
 - row, 9
 - score, 9
- move
 - TTEEntry, 10
- nodes_searched
 - AIStats, 5

PATTERN_FOUR
 types.h, 29
 PATTERN_ONE
 types.h, 29
 PATTERN_THREE
 types.h, 29
 PATTERN_TWO
 types.h, 29
 PATTERN_WIN
 types.h, 29
 print_game_info
 main.c, 51
 pruned
 AIStats, 5

 renderer
 graphics.c, 48
 row
 Move, 9
 VisualMarker, 11
 rule_captures
 GomokuGame, 7
 rule_center_opening
 GomokuGame, 8
 rule_no_double_threes
 GomokuGame, 8

 score
 Move, 9
 TTEnter, 10
 src/ai.c, 31
 src/game.c, 37
 src/graphics.c, 41
 src/main.c, 49

 taken_stones
 GomokuGame, 8
 time_taken
 AIStats, 6
 transposition_table
 ai.c, 36
 TTEnter, 9
 depth, 10
 key, 10
 move, 10
 score, 10
 types.h
 AI_INFINITY, 28
 BLACK, 28
 BOARD_SIZE, 28
 CELL_SIZE, 28
 COLOR_BACKGROUND, 30
 COLOR_BLACK, 30
 COLOR_BLUE, 30
 COLOR_RED, 30
 COLOR_WHITE, 30
 EMPTY, 28
 M_PI, 28
 MAX_MOVES, 28
 MAX_TT_SIZE, 29
 MAX_VISUAL_MARKERS, 29
 PATTERN_FOUR, 29
 PATTERN_ONE, 29
 PATTERN_THREE, 29
 PATTERN_TWO, 29
 PATTERN_WIN, 29
 WHITE, 29
 WINDOW_HEIGHT, 30
 WINDOW_WIDTH, 30

 visual_marker_count
 graphics.c, 49
 visual_markers
 graphics.c, 49
 VisualMarker, 10
 col, 11
 color, 11
 row, 11

 WHITE
 types.h, 29
 window
 graphics.c, 49
 WINDOW_HEIGHT
 types.h, 30
 WINDOW_WIDTH
 types.h, 30