Tetris

Generated by Doxygen 1.13.2

Chapter 1

Tetris

A tetris clone

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 gameBoardType Struct Reference

Data Fields

- int shape_board [BLOCKS_DOWN][BLOCKS_ACROSS]
- int blocked_board [BLOCKS_DOWN][BLOCKS_ACROSS]
- int temp_board [BLOCKS_DOWN][BLOCKS_ACROSS]
- bool hasShapeBeingChosen
- bool GameOver

4.1.1 Detailed Description

ADT of a number of game boards

4.1.2 Field Documentation

4.1.2.1 blocked_board

```
int blocked_board[BLOCKS_DOWN][BLOCKS_ACROSS]
```

Contains occupied squares

4.1.2.2 GameOver

```
bool GameOver
```

Is game finished?

4.1.2.3 hasShapeBeingChosen

```
bool hasShapeBeingChosen
```

Decides whether a new shape is required

4.1.2.4 shape_board

```
int shape_board[BLOCKS_DOWN][BLOCKS_ACROSS]
```

Contains the active shape

4.1.2.5 temp_board

```
int temp_board[BLOCKS_DOWN][BLOCKS_ACROSS]
```

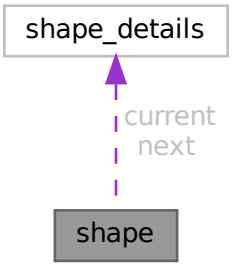
Used to copy the entire board temporarily

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

- shape.c

4.2 shape Struct Reference

Collaboration diagram for shape:



Data Fields

- shape_details **current**
- shape_details **next**
- Vector2 **next_board_shape** [4]
- int **fall_counter**
- int * **ptrToArray**

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

- shape.c

4.3 shape_details Struct Reference

Data Fields

- int **shape_number**
- int **full_details** [4][4]
- int **coordinates** [4]
- int **rotation**
- Color **colour**

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

- shape.c

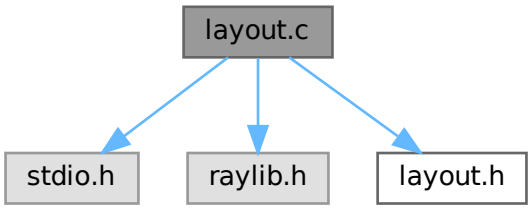
Chapter 5

File Documentation

5.1 layout.c File Reference

```
#include <stdio.h>
#include <raylib.h>
#include "layout.h"
```

Include dependency graph for layout.c:



Functions

- void layoutDrawBoard (void)
- void layoutDrawNextShapeBoard (void)
- void layoutMenu (void)
- int layoutNavigation (int keyPress)

5.1.1 Function Documentation

5.1.1.1 layoutDrawBoard()

```
void layoutDrawBoard (
            void )
```

Draw horizontal and vertical lines of the game board to screen

5.1.1.2 layoutDrawNextShapeBoard()

```
void layoutDrawNextShapeBoard (
            void )
```

Draw horizontal and vertical lines of the next shape board

5.1.1.3 layoutMenu()

```
void layoutMenu (
            void )
```

Draw text for the menu options

5.1.1.4 layoutNavigation()

```
int layoutNavigation (
            int keyPress)
```

Function that returns integer representing menu choice

Parameters

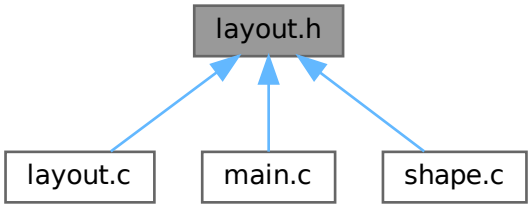
in	*keyPress*	

Return values

<*keyPress*>	integer representing menu choice

5.2 layout.h File Reference

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



Macros

- #define WIDTH 600
- #define HEIGHT 600
- #define SQUARE 20
- #define BOARD_LEFT_POSITION WIDTH ∗ 0.5
- #define BOARD_TOP_POSITION HEIGHT ∗ 0.1
- #define BLOCKS_ACROSS 10
- #define BLOCKS_DOWN 20
- #define MENU_POS_X WIDTH/20
- #define MENU_POS_Y HEIGHT/12
- #define MENU_TXT_SIZE WIDTH/20
- #define NEXT_SHAPE_TOP_LEFT_X BOARD_LEFT_POSITION - (8 ∗ SQUARE)
- #define NEXT_SHAPE_X_CALC(x)
- #define NEXT_SHAPE_Y_CALC(x)

Functions

- void layoutDrawBoard (void)
- void layoutMenu (void)
- int layoutNavigation (int keyPress)

5.2.1 Macro Definition Documentation

5.2.1.1 BLOCKS_ACROSS

```
#define BLOCKS_ACROSS 10
```

How many columns is our gameboard?

5.2.1.2 BLOCKS_DOWN

```
#define BLOCKS_DOWN 20
```

How many rows is our gameboard

5.2.1.3 BOARD_LEFT_POSITION

```
#define BOARD_LEFT_POSITION WIDTH * 0.5
```

Left side of game board as a ratio of width (starting from left side of screen

5.2.1.4 BOARD_TOP_POSITION

```
#define BOARD_TOP_POSITION HEIGHT * 0.1
```

Top position of board as ratio of height (starting from top of the screen)

5.2.1.5 HEIGHT

```
#define HEIGHT 600
```

screen height

5.2.1.6 MENU_POS_X

```
#define MENU_POS_X WIDTH/20
```

Where is our menu text position (x)?

5.2.1.7 MENU_POS_Y

```
#define MENU_POS_Y HEIGHT/12
```

Where is out menu text position (y)?

5.2.1.8 MENU_TXT_SIZE

```
#define MENU_TXT_SIZE WIDTH/20
```

Text size

5.2.1.9 NEXT_SHAPE_TOP_LEFT_X

```
#define NEXT_SHAPE_TOP_LEFT_X BOARD_LEFT_POSITION - (8 * SQUARE)
```

Next shape details. The top left corner of our imaginary next shape board (hidden)

5.2.1.10 NEXT_SHAPE_X_CALC

```
#define NEXT_SHAPE_X_CALC(
              x)
```

Value:

```
((NEXT_SHAPE_TOP_LEFT_X) + (x * SQUARE))
```

Macro returns how many squares across from next shape start point as an integer. Used for placement of each square (x)

5.2.1.11 NEXT_SHAPE_Y_CALC

```
#define NEXT_SHAPE_Y_CALC(
              x)
```

Value:

```
((BOARD_TOP_POSITION) + (x * SQUARE) )
```

Macro returns how many squares down from next shape start point as an integer. Used for placement of each square (y)

5.2.1.12 SQUARE

```
#define SQUARE 20
```

Square size in pixels

5.2.1.13 WIDTH

```
#define WIDTH 600
```

screen width

5.2.2 Function Documentation

5.2.2.1 layoutDrawBoard()

```
void layoutDrawBoard (
            void )
```

Draw horizontal and vertical lines of the game board to screen

5.2.2.2 layoutMenu()

```
void layoutMenu (
            void )
```

Draw text for the menu options

5.2.2.3 layoutNavigation()

```
int layoutNavigation (
            int keyPress)
```

Function that returns integer representing menu choice

Parameters

in	*keyPress*	

Return values

<*keyPress*>	integer representing menu choice

5.3 layout.h

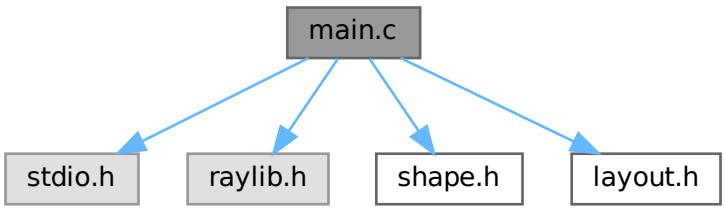
Go to the documentation of this file.

```
00001
00002 #ifndef LAYOUT_H
00003 #define LAYOUT_H
00007 #define WIDTH 600
00008
00011 #define HEIGHT 600
00012
00015 #define SQUARE 20
00016
00021 #define BOARD_LEFT_POSITION     WIDTH * 0.5
00022
00027 #define BOARD_TOP_POSITION      HEIGHT * 0.1
00028
00032 #define BLOCKS_ACROSS   10
00033
00036 #define BLOCKS_DOWN     20
00037
00041 #define MENU_POS_X WIDTH/20
00042
00045 #define MENU_POS_Y HEIGHT/12
00046
00049 #define MENU_TXT_SIZE WIDTH/20
00050
00055 #define NEXT_SHAPE_TOP_LEFT_X   BOARD_LEFT_POSITION - (8 * SQUARE)
00056
00061 #define NEXT_SHAPE_X_CALC(x) ((NEXT_SHAPE_TOP_LEFT_X) + (x * SQUARE))
00062
00067 #define NEXT_SHAPE_Y_CALC(x) ((BOARD_TOP_POSITION) + (x * SQUARE) )
00068
00069 void layoutDrawBoard(void);
00070 //void layoutDrawNextShapeBoard(void);
00071 void layoutMenu(void);
00072 int layoutNavigation(int keyPress);
00073
00074 #endif
```

5.4 main.c File Reference

```
#include <stdio.h>
#include <raylib.h>
#include "shape.h"
#include "layout.h"
```

Include dependency graph for main.c:



Enumerations

- enum gamestate { MENU , START , EXIT }

Functions

- int **main** (void)

5.4.1 Enumeration Type Documentation

5.4.1.1 gamestate

```
enum gamestate
```

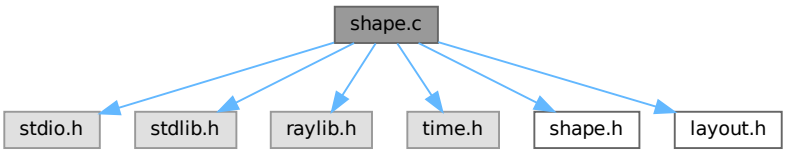
Enumerator

MENU	Tetris remains on menu screen
START	Tetris starts
EXIT	Tetris exits

5.5 shape.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <raylib.h>
#include "time.h"
#include "shape.h"
#include "layout.h"
```

Include dependency graph for shape.c:



Data Structures

- struct gameBoardType
- struct shape_details
- struct shape

Typedefs

- typedef struct gameBoardType gameBoardType
- typedef struct shape_details **shape_details**
- typedef struct shape **shape**

Functions

- gameBoard ShapeInitBoard (void)
- int getNumber (void)
- void generateNextAndActiveShape (gameBoard gameBoard_ptr)
- void transferShapeDetails (int array[ ], int array_full[ ][4], int shape, int rotation)
- Color getColorDetails (int shape)
- void putActiveShapeOnBoard (gameBoard gameBoard_ptr)
- void putOnBlockedBoard (gameBoard gameBoard_ptr)
- void clearScreen (gameBoard gameBoard_ptr)
- void drawActiveShape (gameBoard gameBoard_ptr)
- void drawBlockedShapes (gameBoard gameBoard_ptr)
- void drawNextShape (gameBoard gameBoard_ptr)
- void drawScore (int score)
- void FallActiveShape (gameBoard gameBoard_ptr)
- bool ShapeIsInLeftColumn (void)
- bool ShapeIsInRightColumn (void)
- bool IsAtFloor (gameBoard gameBoard_ptr)
- bool IsOnTop (gameBoard gameBoard_ptr)
- bool IsOnSide (gameBoard gameBoard_ptr)
- int IsRowFull (gameBoard gameBoard_ptr)
- bool destroyRow (gameBoard gameBoard_ptr, int row)
- void MoveLeft (void)
- void MoveRight (void)
- void MoveUp (void)
- bool MoveDown (gameBoard gameBoard_ptr)
- void freeBoard (gameBoard gameBoard_ptr)

Variables

- shape **Shape**

5.5.1 Typedef Documentation

5.5.1.1 gameBoardType

```
typedef struct gameBoardType gameBoardType
```

ADT of a number of game boards

5.5.2 Function Documentation

5.5.2.1 clearScreen()

```
void clearScreen (
            gameBoard gameBoard_ptr)
```

Clear shape from current frame and reset array values for next placement.

5.5.2.2 destroyRow()

```
bool destroyRow (
            gameBoard gameBoard_ptr,
            int row)
```

Cycle through array and for each element that isn't -1, remove that row from the board

Parameters

in	*gameBoard_ptr*	
in	*array*	

5.5.2.3 drawActiveShape()

```
void drawActiveShape (
            gameBoard gameBoard_ptr)
```

Draw active shape onto raylib coordinates using the function DrawRectangle

5.5.2.4 drawBlockedShapes()

```
void drawBlockedShapes (
            gameBoard gameBoard_ptr)
```

Draw blocked squares to screen

Parameters

in	*gameBoard_ptr*	Used for obtaining blocked values from 2D array.

5.5.2.5 drawNextShape()

```
void drawNextShape (
            gameBoard gameBoard_ptr)
```

Draws next shape to the left of game board

Parameters

in	*gameBoard_ptr*	

5.5.2.6 drawScore()

```
void drawScore (
            int score)
```

Draw score to screen

5.5.2.7 FallActiveShape()

```
void FallActiveShape (
            gameBoard gameBoard_ptr)
```

Allows shape to move by moving to the next row down

5.5.2.8 freeBoard()

```
void freeBoard (
            gameBoard gameBoard_ptr)
```

Deallocate memory

5.5.2.9 generateNextAndActiveShape()

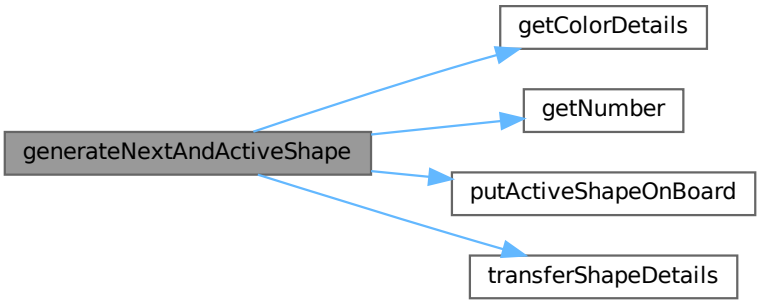
```
void generateNextAndActiveShape (
            gameBoard gameBoard_ptr)
```

This function allocates details to the global variable Shape a variable of type shape_details

Parameters

in	*gameBoard*	Check if a shape has been chosen

Here is the call graph for this function:



5.5.2.10 getColorDetails()

```
Color getColorDetails (
            int shape)
```

Select the appropriate colour for the next shape

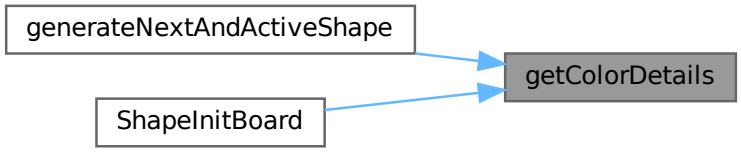
Parameters

in	*shape*	An empty value that requires populating

Return values

<list[shape]>	Returns colour value

Here is the caller graph for this function:



5.5.2.11 getNumber()

```
int getNumber (
            void )
```

Returns

Number used in the generation of next random shape

Here is the caller graph for this function:

5.5.2.12 IsAtFloor()

```
bool IsAtFloor (
            gameBoard gameBoard_ptr)
```

Checks if any square of the moving shape is one the bottom row

Parameters

in	*gameBoard_ptr*	

5.5.2.13 IsOnSide()

```
bool IsOnSide (
            gameBoard gameBoard_ptr)
```

Function to check if there is a block to left side of active shape

5.5.2.14 IsOnTop()

```
bool IsOnTop (
            gameBoard gameBoard_ptr)
```

Function that checks position of shape. If there is a blocked shape underneath return true.

Parameters

in	*gameBoard_ptr*	

5.5.2.15 IsRowFull()

```
int IsRowFull (
            gameBoard gameBoard_ptr)
```

If a row is full of blocked tiles, add that row number to array

Parameters

in	*gameBoard_ptr*	
out	*array*	array containing 4 elements set to -1
out	*pointer*	to variable score

5.5.2.16 MoveDown()

```
bool MoveDown (
            gameBoard gameBoard_ptr)
```

Not yet used

5.5.2.17 MoveLeft()

```
void MoveLeft (
            void )
```

Move object left

5.5.2.18 MoveRight()

```
void MoveRight (
            void )
```

Move object right

5.5.2.19 MoveUp()

```
void MoveUp (
            void )
```

Changes current rotation value of moving shape by altering global variable member Shape.current

5.5.2.20 putActiveShapeOnBoard()

```
void putActiveShapeOnBoard (
            gameBoard gameBoard_ptr)
```

Function that moves pointer a specified distance four times using Shape.current.coordinates to place a '1' on the shape_board to signal a moving block

Parameters

out	*gameBoard_ptr*	Accesses four elements of the member shape_board

Here is the caller graph for this function:

5.5.2.21 putOnBlockedBoard()

```
void putOnBlockedBoard (
            gameBoard gameBoard_ptr)
```

Function that moves a pointer a specified distance four times using Shape.current.coordinates to place a '1' on the blocked_board to signal an occupied space. Takes moving shape off the board and converts it to a blocked shape. Signals that a current shape is no longer in existence.

Parameters

out	*gameBoard_ptr*	Accesses four elements of the member blocked_board

5.5.2.22 ShapeInitBoard()

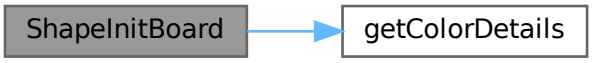
```
gameBoard ShapeInitBoard (
            void )
```

Allocate space for a pointer to type gameBoard and initialise some values.

Return values

<gameBoard_ptr>	Returns a pointer of type gameBoard

Here is the call graph for this function:



5.5.2.23 ShapeIsInLeftColumn()

```
bool ShapeIsInLeftColumn (
            void )
```

Checks the current shape position for its shape positions returns true if at edge

5.5.2.24 ShapeIsInRightColumn()

```
bool ShapeIsInRightColumn (
            void )
```

Checks the current shape position for its shape positions returns true if at edge

5.5.2.25 transferShapeDetails()

```
void transferShapeDetails (
            int array[],
            int array_full[][4],
            int shape,
            int rotation)
```

Transfer shape details (coordinates etc) to supplied arguments

Parameters

out	*array*	Where are we sending the current shape array data?
out	*array_full*	Where are we sending the full shape data (each rotation)?
out	*shape*	The shape that is in use
out	*rotation*	Used to iterate onto next rotation when Up arrow is pressed

Here is the caller graph for this function:

5.6 shape.h

```
00001
00002 #ifndef SHAPE_H
00003 #define SHAPE_H
00004
00005 typedef struct gameBoardType *gameBoard;
00006 gameBoard ShapeInitBoard(void);
00007
00008 int getNumber(void);
00009 void generateNextAndActiveShape(gameBoard gameBoard_ptr);
00010 void transferShapeDetails(int array[], int array_full[][4], int shape, int rotation);
00011 Color getColorDetails(int shape);
00012
00013 void putActiveShapeOnBoard(gameBoard gameBoard_ptr);
00014
00015 void putOnBlockedBoard(gameBoard gameBoard_ptr);
00016
00017 void drawActiveShape(gameBoard gameBoard_ptr);
00018 void drawBlockedShapes(gameBoard gameBoard_ptr);
00019 void drawNextShape(gameBoard gameBoard_ptr);
00020
00021 void drawScore(int score);
00022
00023 void clearScreen(gameBoard gameBoard_ptr);
00024 void FallActiveShape(gameBoard gameBoard_ptr);
00025
00026 bool ShapeIsInLeftColumn(void);
00027 bool ShapeIsInRightColumn(void);
00028 bool IsAtFloor(gameBoard gameBoard_ptr);
00029 bool IsOnTop(gameBoard gameBoard_ptr);
00030 bool IsOnSide(gameBoard gameBoard_ptr);
00031
00032 int IsRowFull(gameBoard gameBoard_ptr);
00033 bool destroyRow(gameBoard gameBoard_ptr, int row);
00034 static bool FallBlocked(gameBoard gameBoard_ptr, int i);
00035
00036 void MoveLeft(void);
00037 void MoveRight(void);
00038 void MoveUp(void);
00039 bool MoveDown(gameBoard gameBoard_ptr);
00040
00041 void freeBoard(gameBoard gameBoard_ptr);
00042
00043 #endif
00044
00045
```

Index