

Tetris

Generated by Doxygen 1.13.2



<b>1 Tetris</b>	<b>1</b>
<b>2 Data Structure Index</b>	<b>3</b>
2.1 Data Structures	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Data Structure Documentation</b>	<b>7</b>
4.1 gameBoardType Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 blocked_board	7
4.1.2.2 GameOver	7
4.1.2.3 hasShapeBeingChosen	7
4.1.2.4 shape_board	8
4.1.2.5 temp_board	8
4.2 shape Struct Reference	8
4.3 shape_details Struct Reference	9
<b>5 File Documentation</b>	<b>11</b>
5.1 layout.c File Reference	11
5.1.1 Function Documentation	11
5.1.1.1 layoutDrawBoard()	11
5.1.1.2 layoutDrawNextShapeBoard()	12
5.1.1.3 layoutMenu()	12
5.1.1.4 layoutNavigation()	12
5.2 layout.h File Reference	12
5.2.1 Macro Definition Documentation	13
5.2.1.1 BLOCKS_ACROSS	13
5.2.1.2 BLOCKS_DOWN	13
5.2.1.3 BOARD_LEFT_POSITION	13
5.2.1.4 BOARD_TOP_POSITION	13
5.2.1.5 HEIGHT	14
5.2.1.6 MENU_POS_X	14
5.2.1.7 MENU_POS_Y	14
5.2.1.8 MENU_TXT_SIZE	14
5.2.1.9 NEXT_SHAPE_TOP_LEFT_X	14
5.2.1.10 NEXT_SHAPE_X_CALC	14
5.2.1.11 NEXT_SHAPE_Y_CALC	14
5.2.1.12 SQUARE	15
5.2.1.13 WIDTH	15
5.2.2 Function Documentation	15
5.2.2.1 layoutDrawBoard()	15

5.2.2.2 layoutMenu()	15
5.2.2.3 layoutNavigation()	15
5.3 layout.h	16
5.4 main.c File Reference	16
5.4.1 Enumeration Type Documentation	17
5.4.1.1 gamestate	17
5.5 shape.c File Reference	17
5.5.1 Typedef Documentation	18
5.5.1.1 gameBoardType	18
5.5.2 Function Documentation	18
5.5.2.1 clearScreen()	18
5.5.2.2 destroyRow()	18
5.5.2.3 drawActiveShape()	19
5.5.2.4 drawBlockedShapes()	19
5.5.2.5 drawNextShape()	19
5.5.2.6 drawScore()	19
5.5.2.7 FallActiveShape()	19
5.5.2.8 freeBoard()	20
5.5.2.9 generateNextAndActiveShape()	20
5.5.2.10 getColorDetails()	20
5.5.2.11 getNumber()	21
5.5.2.12 IsAtFloor()	21
5.5.2.13 IsOnSide()	22
5.5.2.14 IsOnTop()	22
5.5.2.15 IsRowFull()	22
5.5.2.16 MoveDown()	22
5.5.2.17 MoveLeft()	22
5.5.2.18 MoveRight()	23
5.5.2.19 MoveUp()	23
5.5.2.20 putActiveShapeOnBoard()	23
5.5.2.21 putOnBlockedBoard()	23
5.5.2.22 ShapeInitBoard()	23
5.5.2.23 ShapesInLeftColumn()	24
5.5.2.24 ShapesInRightColumn()	24
5.5.2.25 transferShapeDetails()	24
5.6 shape.h	25
<b>Index</b>	<b>27</b>

# Chapter 1

## Tetris

A tetris clone



## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">gameBoardType</a>	7
<a href="#">shape</a>	8
<a href="#">shape_details</a>	9





# Chapter 3

## File Index

### 3.1 File List

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

<a href="#">layout.c</a>	11
<a href="#">layout.h</a>	12
<a href="#">main.c</a>	16
<a href="#">shape.c</a>	17
<a href="#">shape.h</a>	25



## 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]
```

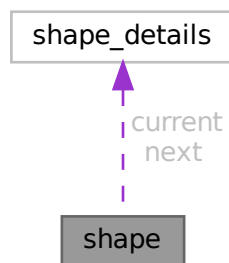
Used to copy the entire board temporarily

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](#)

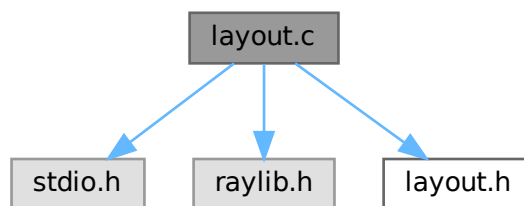


## Chapter 5

# 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

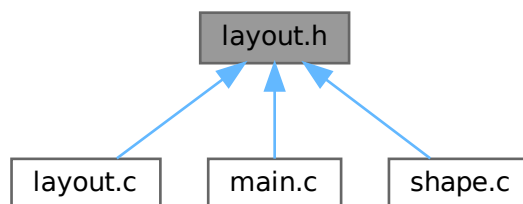
in	<i>keyPress</i>	
----	-----------------	--

##### Return values

< <i>keyPress</i> >	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	<i>keyPress</i>	
----	-----------------	--

#### Return values

< <i>keyPress</i> >	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

```

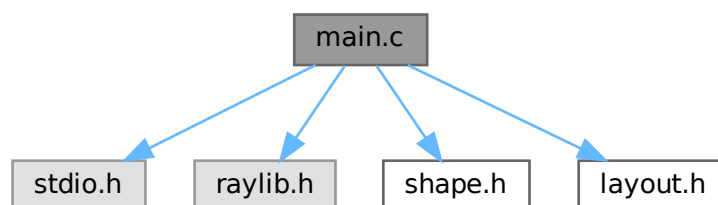
## 5.4 main.c File Reference

```

#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](#)

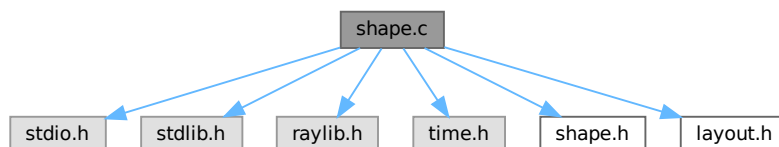
#### Enumerator

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 [ShapelsInLeftColumn](#) (void)
- bool [ShapelsInRightColumn](#) (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	<i>gameBoard_ptr</i>	
in	<i>array</i>	

**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	<i>gameBoard_ptr</i>	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	<i>gameBoard_ptr</i>	
----	----------------------	--

**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()

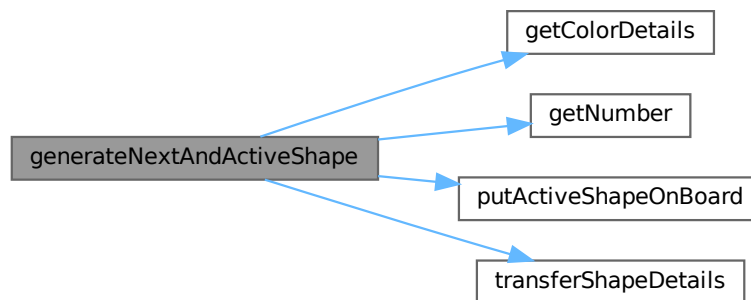
```
void generateNextAndActiveShape (
    gameBoard gameBoard_ptr)
```

This function allocates details to the global variable Shape a variable of type *shape\_details*

#### Parameters

in	<i>gameBoard</i>	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	<i>shape</i>	An empty value that requires populating
----	--------------	-----------------------------------------

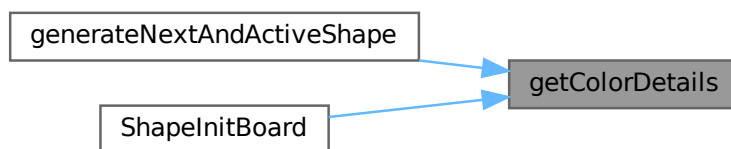
#### Return values

< <i>list[shape]</i> >	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	<code>gameBoard_ptr</code>	
----	----------------------------	--

### 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	<i>gameBoard_ptr</i>	
----	----------------------	--

### 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	<i>gameBoard_ptr</i>	
out	<i>array</i>	array containing 4 elements set to -1
out	<i>pointer</i>	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

< <i>gameBoard_ptr</i> >	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	<i>array</i>	Where are we sending the current shape array data?
out	<i>array_full</i>	Where are we sending the full shape data (each rotation)?
out	<i>shape</i>	The shape that is in use
out	<i>rotation</i>	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

- blocked\_board
  - gameBoardType, [7](#)
- BLOCKS\_ACROSS
  - layout.h, [13](#)
- BLOCKS\_DOWN
  - layout.h, [13](#)
- BOARD\_LEFT\_POSITION
  - layout.h, [13](#)
- BOARD\_TOP\_POSITION
  - layout.h, [13](#)
- clearScreen
  - shape.c, [18](#)
- destroyRow
  - shape.c, [18](#)
- drawActiveShape
  - shape.c, [19](#)
- drawBlockedShapes
  - shape.c, [19](#)
- drawNextShape
  - shape.c, [19](#)
- drawScore
  - shape.c, [19](#)
- EXIT
  - main.c, [17](#)
- FallActiveShape
  - shape.c, [19](#)
- freeBoard
  - shape.c, [19](#)
- gameBoardType, [7](#)
  - blocked\_board, [7](#)
  - GameOver, [7](#)
  - hasShapeBeingChosen, [7](#)
  - shape.c, [18](#)
  - shape\_board, [7](#)
  - temp\_board, [8](#)
- GameOver
  - gameBoardType, [7](#)
- gamestate
  - main.c, [17](#)
- generateNextAndActiveShape
  - shape.c, [20](#)
- getColorDetails
  - shape.c, [20](#)
- getNumber
  - shape.c, [21](#)
- hasShapeBeingChosen
  - gameBoardType, [7](#)
- HEIGHT
  - layout.h, [13](#)
- IsAtFloor
  - shape.c, [21](#)
- IsOnSide
  - shape.c, [21](#)
- IsOnTop
  - shape.c, [22](#)
- IsRowFull
  - shape.c, [22](#)
- layout.c, [11](#)
  - layoutDrawBoard, [11](#)
  - layoutDrawNextShapeBoard, [11](#)
  - layoutMenu, [12](#)
  - layoutNavigation, [12](#)
- layout.h, [12](#)
  - BLOCKS\_ACROSS, [13](#)
  - BLOCKS\_DOWN, [13](#)
  - BOARD\_LEFT\_POSITION, [13](#)
  - BOARD\_TOP\_POSITION, [13](#)
  - HEIGHT, [13](#)
  - layoutDrawBoard, [15](#)
  - layoutMenu, [15](#)
  - layoutNavigation, [15](#)
  - MENU\_POS\_X, [14](#)
  - MENU\_POS\_Y, [14](#)
  - MENU\_TXT\_SIZE, [14](#)
  - NEXT\_SHAPE\_TOP\_LEFT\_X, [14](#)
  - NEXT\_SHAPE\_X\_CALC, [14](#)
  - NEXT\_SHAPE\_Y\_CALC, [14](#)
  - SQUARE, [14](#)
  - WIDTH, [15](#)
- layoutDrawBoard
  - layout.c, [11](#)
  - layout.h, [15](#)
- layoutDrawNextShapeBoard
  - layout.c, [11](#)
- layoutMenu
  - layout.c, [12](#)
  - layout.h, [15](#)
- layoutNavigation
  - layout.c, [12](#)
  - layout.h, [15](#)
- main.c, [16](#)
  - EXIT, [17](#)

- gamestate, [17](#)
- MENU, [17](#)
- START, [17](#)
- MENU
  - main.c, [17](#)
- MENU\_POS\_X
  - layout.h, [14](#)
- MENU\_POS\_Y
  - layout.h, [14](#)
- MENU\_TXT\_SIZE
  - layout.h, [14](#)
- MoveDown
  - shape.c, [22](#)
- MoveLeft
  - shape.c, [22](#)
- MoveRight
  - shape.c, [22](#)
- MoveUp
  - shape.c, [23](#)
- NEXT\_SHAPE\_TOP\_LEFT\_X
  - layout.h, [14](#)
- NEXT\_SHAPE\_X\_CALC
  - layout.h, [14](#)
- NEXT\_SHAPE\_Y\_CALC
  - layout.h, [14](#)
- putActiveShapeOnBoard
  - shape.c, [23](#)
- putOnBlockedBoard
  - shape.c, [23](#)
- shape, [8](#)
- shape.c, [17](#)
  - clearScreen, [18](#)
  - destroyRow, [18](#)
  - drawActiveShape, [19](#)
  - drawBlockedShapes, [19](#)
  - drawNextShape, [19](#)
  - drawScore, [19](#)
  - FallActiveShape, [19](#)
  - freeBoard, [19](#)
  - gameBoardType, [18](#)
  - generateNextAndActiveShape, [20](#)
  - getColorDetails, [20](#)
  - getNumber, [21](#)
  - IsAtFloor, [21](#)
  - IsOnSide, [21](#)
  - IsOnTop, [22](#)
  - IsRowFull, [22](#)
  - MoveDown, [22](#)
  - MoveLeft, [22](#)
  - MoveRight, [22](#)
  - MoveUp, [23](#)
  - putActiveShapeOnBoard, [23](#)
  - putOnBlockedBoard, [23](#)
  - ShapelInitBoard, [23](#)
  - ShapelsInLeftColumn, [24](#)
  - ShapelsInRightColumn, [24](#)
  - transferShapeDetails, [24](#)
- shape\_board
  - gameBoardType, [7](#)
- shape\_details, [9](#)
- ShapelInitBoard
  - shape.c, [23](#)
- ShapelsInLeftColumn
  - shape.c, [24](#)
- ShapelsInRightColumn
  - shape.c, [24](#)
- SQUARE
  - layout.h, [14](#)
- START
  - main.c, [17](#)
- temp\_board
  - gameBoardType, [8](#)
- Tetris, [1](#)
- transferShapeDetails
  - shape.c, [24](#)
- WIDTH
  - layout.h, [15](#)