

2048 EDD

Generated by Doxygen 1.8.6

Mon Apr 20 2015 18:19:31

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	game Struct Reference	5
3.2	grid_s Struct Reference	5
3.3	strategy_s Struct Reference	5
3.3.1	Detailed Description	6
3.3.2	Field Documentation	6
3.3.2.1	free_strategy	6
3.3.2.2	name	6
3.3.2.3	play_move	6
3.4	vars_draw Struct Reference	6
4	File Documentation	7
4.1	include/grid.h File Reference	7
4.1.1	Detailed Description	8
4.1.2	Macro Definition Documentation	8
4.1.2.1	GRID_SIDE	8
4.1.3	Typedef Documentation	8
4.1.3.1	grid	8
4.1.4	Function Documentation	8
4.1.4.1	add_tile	8
4.1.4.2	can_move	8
4.1.4.3	copy_grid	9
4.1.4.4	delete_grid	9
4.1.4.5	do_move	9
4.1.4.6	game_over	9
4.1.4.7	get_tile	9

4.1.4.8	grid_score	10
4.1.4.9	new_grid	10
4.1.4.10	play	10
4.1.4.11	set_tile	10
4.2	include/strategy.h File Reference	11
4.2.1	Detailed Description	11
4.2.2	Typedef Documentation	11
4.2.2.1	strategy	11
4.2.3	Function Documentation	11
4.2.3.1	FirstStrat	11
4.2.3.2	free_memless_strat	12
Index		13

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

game	5
grid_s	5
strategy_s	5
vars_draw	6

Chapter 2

File Index

2.1 File List

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

include/ afficher.h	??
include/ grid.h	
Contains structures and functions needed to play 2048 game	7
include/ keyboard.h	??
include/ Keybord.h	??
include/ strategy.h	
Defines the "strategy" structure	11

Chapter 3

Data Structure Documentation

3.1 game Struct Reference

Data Fields

- state **st**
- bool **disable_play_mouse**
- Uint32 **time**
- int **fps**
- [grid](#) **g**

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

- [src/2048-gui.c](#)

3.2 grid_s Struct Reference

Data Fields

- [tile](#) **g** [[GRID_SIDE](#)][[GRID_SIDE](#)]
- unsigned long int **score**

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

- [src/grid.c](#)

3.3 strategy_s Struct Reference

```
#include <strategy.h>
```

Data Fields

- void * **mem**
- char * [name](#)
- [dir](#)(* [play_move](#))(strategy, [grid](#))
- void(* [free_strategy](#))(strategy)

3.3.1 Detailed Description

Structure that contains a strategy.

A strategy is essentially a function (`play_move`) which, given a partially filled grid, picks the next direction to be played. More advanced strategies may require a history of previously played moves (or even games) ; in such cases, a strategy may use the `<mem>` field to store such information.

3.3.2 Field Documentation

3.3.2.1 `void(* free_strategy)(strategy)`

A function which returns the direction chosen by the strategy. The first parameter is a pointer to the current strategy (useless for memoryless strategies). The second one is the current grid.

3.3.2.2 `char* name`

Points to the data stored by the strategy.

3.3.2.3 `dir(* play_move)(strategy, grid)`

The strategy's name. This will be used to display the tournament scores.

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

- [include/strategy.h](#)

3.4 vars_draw Struct Reference

Data Fields

- `SDL_Surface *` **screen**
- `TTF_Font **` **fonts**

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

- `src/2048-gui.c`

Chapter 4

File Documentation

4.1 include/grid.h File Reference

Contains structures and functions needed to play 2048 game.

```
#include <stdbool.h>
```

Macros

- #define `GRID_SIDE` 4

Typedefs

- typedef struct `grid_s` * `grid`
Contain game's status: tiles and current score.
- typedef unsigned int `tile`
*Log₂-encoded tile : 0 is empty, i is 2**i.*
- typedef enum `dir_e` `dir`
List of accepted movement in the game.

Enumerations

- enum `dir_e` { `UP`, `LEFT`, `DOWN`, `RIGHT` }
List of accepted movement in the game.

Functions

- `grid new_grid` ()
Initialize grid structure.
- void `delete_grid` (`grid` g)
Destroy the grid and free allocated memory.
- void `copy_grid` (`grid` src, `grid` dst)
Clone the grid.
- unsigned long int `grid_score` (`grid` g)
Get game's score.
- `tile get_tile` (`grid` g, int x, int y)

- Get tile (log₂-encoded) from the grid by specifying his coordinates.*

 - void `set_tile` (`grid` g, int x, int y, `tile` t)

Change tile's value.
- bool `can_move` (`grid` g, `dir` d)

Verify if a given movement is possible.
- bool `game_over` (`grid` g)

Verify game's status, if no more movement is possible the game is over.
- void `do_move` (`grid` g, `dir` d)

Move every tiles of the grid in the direction specified by the user.
- void `add_tile` (`grid` g)

Randomly add a tile in the grid in a free space when a movement is finished. With probability 9/10 the new tile has value 2 and with probability 1/10 the new tile has value 4.
- void `play` (`grid` g, `dir` d)

Play a direction in the grid.

4.1.1 Detailed Description

Contains structures and functions needed to play 2048 game.

4.1.2 Macro Definition Documentation

4.1.2.1 `#define GRID_SIDE 4`

Grid dimension

4.1.3 Typedef Documentation

4.1.3.1 `typedef struct grid_s* grid`

Contain game's status: tiles and current score.

```
X 0 1 ... GRID_SIDE-1 +-+ +-+ ... +-+ | | | ... + | 0 +-+ +-+ ... +-+ | | | ... | 1 +-+ +-+ ... +-+ Y ... .. +-+ +-+ ...
+-+ | | | ... | | GRID_SIDE-1 +-+ +-+ ... +-+
```

4.1.4 Function Documentation

4.1.4.1 `void add_tile (grid g)`

Randomly add a tile in the grid in a free space when a movement is finished. With probability 9/10 the new tile has value 2 and with probability 1/10 the new tile has value 4.

Parameters

<code>g</code>	the grid
----------------	----------

Precondition

`grid` g must contain at least one empty tile.

4.1.4.2 `bool can_move (grid g, dir d)`

Verify if a given movement is possible.

Parameters

<i>g</i>	the grid
<i>d</i>	movement's direction

Returns

true if the movement is possible, false else

4.1.4.3 void copy_grid (grid *src*, grid *dst*)

Clone the grid.

Parameters

<i>src</i>	the grid to copy
<i>dst</i>	the copied grid

4.1.4.4 void delete_grid (grid *g*)

Destroy the grid and free allocated memory.

Parameters

<i>g</i>	the grid to destroy
----------	---------------------

4.1.4.5 void do_move (grid *g*, dir *d*)

Move every tiles of the grid in the direction specified by the user.

Parameters

<i>g</i>	the grid
<i>d</i>	the chosen direction

Precondition

the movement *d* must be possible (i.e. can_move(*g*,*d*) == true).

4.1.4.6 bool game_over (grid *g*)

Verify game's status, if no more movement is possible the game is over.

Parameters

<i>g</i>	the grid
----------	----------

Returns

true if there is no more possible movements, false else

4.1.4.7 tile get_tile (grid *g*, int *x*, int *y*)

Get tile (log₂-encoded) from the grid by specifying his coordinates.

Parameters

<i>g</i>	the grid
<i>x</i>	and <i>y</i> tile's coordinates

Returns

the tile

Precondition

$0 \leq x < \text{GRID_SIDE}$ and $0 \leq y < \text{GRID_SIDE}$

4.1.4.8 unsigned long int grid_score (grid *g*)

Get game's score.

Parameters

<i>g</i>	the grid
----------	----------

Returns

the computed score during the game

4.1.4.9 grid new_grid ()

Initialize grid structure.

Returns

created an empty grid with score equal to 0

4.1.4.10 void play (grid *g*, dir *d*)

Play a direction in the grid.

Parameters

<i>g</i>	the grid
<i>d</i>	the direction

Precondition

the movement *d* must be possible (i.e. `can_move(g,d) == true`).

4.1.4.11 void set_tile (grid *g*, int *x*, int *y*, tile *t*)

Change tile's value.

Parameters

<i>g</i>	the grid
<i>x</i>	and y tile's coordinates
<i>t</i>	new tile's value

4.2 include/strategy.h File Reference

Defines the "strategy" structure.

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

Data Structures

- struct [strategy_s](#)

Typedefs

- typedef struct [strategy_s](#) * [strategy](#)

Functions

- [strategy firstStratConstruct](#) ()
- void [free_memless_strat](#) ([strategy](#) strat)
- [dir FirstStrat](#) ([strategy](#) s, [grid](#) g)
choose the first direction possible in this order: LEFT, DOWN, UP, RIGHT

Variables

- [strategy](#)(* [listFunctionsStrat](#) [])()
- char * [listNamesStrat](#) []

4.2.1 Detailed Description

Defines the "strategy" structure. 1.0

4.2.2 Typedef Documentation

4.2.2.1 typedef struct [strategy_s](#)* [strategy](#)

strategy is a pointer to a [strategy_s](#) structure

4.2.3 Function Documentation

4.2.3.1 [dir FirstStrat](#) ([strategy](#) s, [grid](#) g)

choose the first direction possible in this order: LEFT, DOWN, UP, RIGHT

Parameters

<i>strategy</i>	s, a structure strategy
<i>grid</i>	g, the grid

4.2.3.2 void free_memless_strat (strategy strat)

Naively frees the <strat> pointer.

Index

add_tile
 grid.h, 8

can_move
 grid.h, 8

copy_grid
 grid.h, 9

delete_grid
 grid.h, 9

do_move
 grid.h, 9

FirstStrat
 strategy.h, 11

free_memless_strat
 strategy.h, 12

free_strategy
 strategy_s, 6

GRID_SIDE
 grid.h, 8

game, 5

game_over
 grid.h, 9

get_tile
 grid.h, 9

grid
 grid.h, 8

grid.h
 add_tile, 8
 can_move, 8
 copy_grid, 9
 delete_grid, 9
 do_move, 9
 GRID_SIDE, 8
 game_over, 9
 get_tile, 9
 grid, 8
 grid_score, 10
 new_grid, 10
 play, 10
 set_tile, 10

grid_s, 5

grid_score
 grid.h, 10

include/grid.h, 7

include/strategy.h, 11

name

 strategy_s, 6

new_grid
 grid.h, 10

play
 grid.h, 10

play_move
 strategy_s, 6

set_tile
 grid.h, 10

strategy
 strategy.h, 11

strategy.h
 FirstStrat, 11
 free_memless_strat, 12
 strategy, 11

strategy_s, 5
 free_strategy, 6
 name, 6
 play_move, 6

vars_draw, 6