Game of Life by John Conway

Generated by Doxygen 1.9.4

Chapter 1

Conway's Game of Life implementation by Lennard Wittenberg in the language C

In this project contains my attempt at the project Game of Life by John Conway. written in C and complied with make.

1.1 What is the Game of Life short explenation

This is the describtion by the Cornell University
The Game of Life (an example of a cellular automaton) is played on an infinite two-dimensional rectangular grid of cells. Each cell can be either alive or dead. The status of each cell changes each turn of the game (also called a generation) depending on the statuses of that cell's 8 neighbors.

1.2 current project state

date 25.03.2024
The core functionality of the Game of Life is in working order. The simulation is behaving as expected. All files compile without any errors, and the code coverage is 100 percent. The simulation is currently being visualized through console printing. Therefore, a possible improvement could be to use SDL2 instead. However, as of this moment, there are no plans on making those changes.

1.3 How access and use the project.

After downloading the project. Open a new terminal and
1.) Enter the project folder with the cd command.
2.) Input the make command into the terminal
3.) Enter the new release folder with the cd command. cd build/release/
4.) Enter ./game_of_life and than a series of number pairs like 0 0 0 1 1 1 two numbers form a pair and it is important to leave spaces between each number.
note: The rules and specialties of this project are detailed in the file placerholder.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 board Struct Reference

Public Attributes

- int $**$ **current_cells**
- int $**$ **updated_cells**
- int **max_rows**
- int **max_cols**

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

- gol_board.h

4.2 cmdargs Struct Reference

```
#include <cmdargs.h>
```

Public Attributes

- int $**$ **a**
- int **max_rows**

4.2.1 Detailed Description

Holds all the possible command line arguments

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

- cmdargs.h

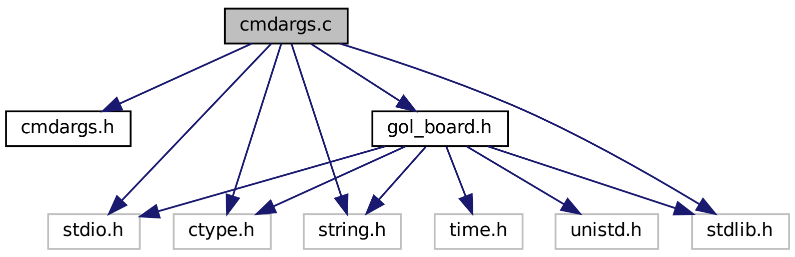
Chapter 5

File Documentation

5.1 cmdargs.c File Reference

```
#include "cmdargs.h"
#include "gol_board.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

Include dependency graph for cmdargs.c:



Functions

- int cmdargs_parse (struct cmdargs ∗cmdargs, int argc, char ∗argv[ ], int r, int c)
- int **argparse_int** (char ∗str)
- void **delete_buffer** (struct cmdargs game)

5.1.1 Function Documentation

5.1.1.1 cmdargs_parse()

```
int cmdargs_parse (
            struct cmdargs * cmdargs,
            int argc,
            char * argv[ ],
            int r,
            int c )
```

Parse the given command line arguments into the cmdargs struct.

Parameters

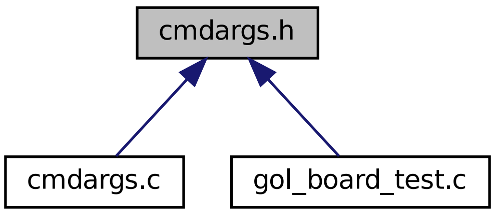
cmdargs	
argc	the size of argv as supplied in main().
argv	the the array of arguments as supplied in main().

Returns

1 on success, 0 on failure.

5.2 cmdargs.h File Reference

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



Classes

- struct cmdargs

Functions

- int cmdargs_parse (struct cmdargs ∗cmdargs, int argc, char ∗argv[ ], int r, int c)
- int **argparse_int** (char ∗str)
- void **delete_buffer** (struct cmdargs game)

5.2.1 Function Documentation

5.2.1.1 cmdargs_parse()

```
int cmdargs_parse (
            struct cmdargs * cmdargs,
            int argc,
            char * argv[ ],
            int r,
            int c )
```

Parse the given command line arguments into the cmdargs struct.

Parameters

cmdargs	
argc	the size of argv as supplied in main().
argv	the the array of arguments as supplied in main().

Returns

1 on success, 0 on failure.

5.3 cmdargs.h

Go to the documentation of this file.

```c

#ifndef _CMDARGS_H_
#define _CMDARGS_H_

struct cmdargs
{
    int **a; // array with two dimensions to store the coordinates of initialized live cells.
        // The first dimension represents the y-axis and the second dimension represents the x-axis
    int max_rows;
};

int cmdargs_parse(struct cmdargs *cmdargs, int argc, char *argv[], int r, int c);
//
int argparse_int(char *str);
//
void delete_buffer(struct cmdargs game);
#endif
```

5.4 gol_board.h

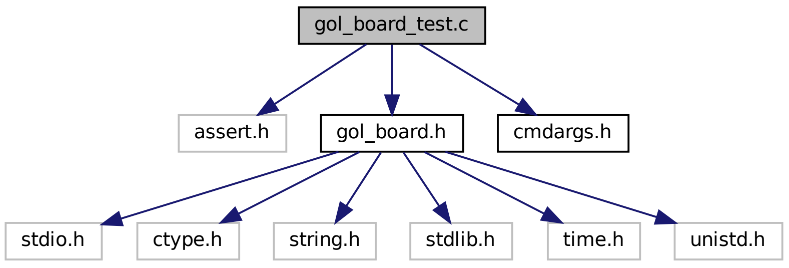
```c
#ifndef GOL_BOARD_H
#define GOL_BOARD_H

#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
#include<unistd.h>

struct board{
```

```
// @param cells represents the cells in the Game of Life.  the array should only hold 1 --> alive or 0
   --> dead.
// other datatypes like bool or char could be used instead.
// The first dimension is supposed to represent the rows of the game_grid.
// The second dimension is supposed to represent the coloums of the game_grid.
// @param current_cells represent the current generation
// @param updated_cells represents the next generation.
    int **current_cells;
    int **updated_cells;
    int max_rows;
    int max_cols;
};
//
/*
* Initializing the game grid in the form of a coordinate system.
* The top left corner has the coordinates x = 0; y = 0
* The top right corner has the coordinates x = max x; y = 0
* The bottom left corner has the coordinates x = 0; y = max y
* The bottom right corner has the coordinates x = max x; y = max y
* At the start of the game all cells are dead.
*@param r and c represent the maximum values for the rows and coloumns.
*/
struct board initialize_board(struct board game, int r, int c);
//
/*
* Support function to test if a specific cell in inside of the grid.
* used for the count_live_neighbours function and indirectly for the enforce_rules function.
*/
int cell_in_grid(struct board game, int cur_row, int cur_col);
//
/*
* Support function that counts all surrounding live neighbours of a inspected cell.
* @param cur_row and cur_col represent the x and y coordinates of the inspected cell.
*/
int count_live_neighbours(struct board game, int cur_row, int cur_col);
//
/*
* Function to apply the rules of Conways game of Life.
*/
void enforce_rules(struct board game);
//
/*
* Function to free the allocated memory of the structur
*/
void delete_board(struct board game);
//
/*
* Function to print out the live cells of the grid to create the Game of Life simulation.
*/
void print_game(struct board game);
//
/*
* transfering the data of updated_cells into current_cells and declaring all cells in updated cells as
   dead.
*/
void swap_cell_states(struct board game);
//
/*
* Function to varify if any live cells are left withing the game.
*/
int check_dead_life(struct board game);
#endif
```

5.5 gol_board_test.c File Reference

```
#include <assert.h>
#include "gol_board.h"
#include "cmdargs.h"
```

Include dependency graph for gol_board_test.c:



Functions

- int main (int argc, char ∗∗argv)

5.5.1 Detailed Description

Test for functions in gcd_functions.c

5.5.2 Function Documentation

5.5.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Main entry for the test.

Index