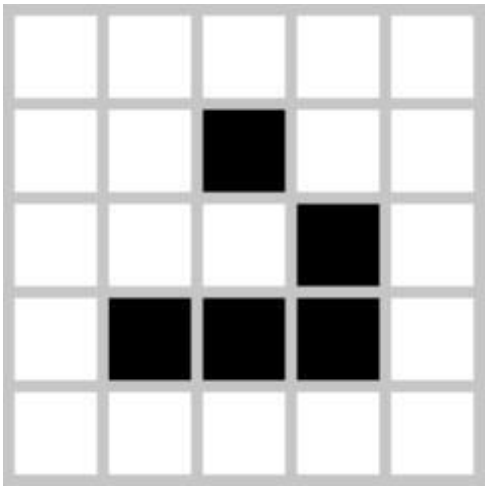

Game of Life

Lothar Gomoluch, Oliver Röckener und Niko Tepe

Version 1.0

Montag, 22.06.2020

File Index

File List

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

buffer.h .....
game.h .....
main.c .....
menu.h .....

Data Structure Documentation

cell Struct Reference

Data Fields

int **alive**
int **livingNeighbors**
struct cell * neighborCell [8]

menu_button Struct Reference

Data Fields

char **label** [20]
COORD **pos**

settings Struct Reference

Data Fields

COORD **gridsize**
char **symbolAlive**
char **symbolDead**
int **generationsToCalc**
int **iterationsPerSecond**
COORD **hud_currentGeneration_pos**
COORD **hud_gridSize_pos**
COORD **hud_generationsToCalc_pos**
COORD **hud_iterationsPerSecond_pos**
COORD **hud_aliveCells_pos**
COORD **hud_shortcutInfo_pos**

File Documentation

main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "include/menu.h"
#include "include/game.h"
#include "include/buffer.h"
```

Functions

void **run** (int generationsToCalc, int ticksPerSecond)
void **tick** (int *end_game, int *pause_game)
Diese Funktion berechnet die nächste Iteration des GOL.

void **settings_menu** ()
generiert ein Untermenü: Settingsmenu

void **main_menu** ()
generiert das Hauptmenü

void **init_settings** ()
initialisiert Grundwerte für die Programmausführung

int **main** ()
void **start_game** (int is_random)
diese Funktion startet ein Spiel

void **start_menu** ()
generiert ein Untermenü: startmenu

void **rule_menu** ()
generiert ein Untermenü: Rulemenu

Variables

struct **settings gamesettings**
struct rule **gamerules**
struct **cell** ** **grid**
struct **cell** ** **gridcopy**
char * **buffer**
struct **menu_button mainMenu_Button** [4]
struct **menu_button startMenu_Button** [2]
struct **menu_button settingsMenu_Button** [6]
struct **menu_button ruleMenu_Button** [3]
int **aliveCells** = 0
int **aliveCellsPrevGen** = 0
int **currentGeneration** = 0

int **iterationsSinceLastChange** = 0
int **runtime_start** = 0

Function Documentation

void init_settings ()

initialisiert Grundwerte für die Programmausführung

void main_menu ()

generiert das Hauptmenü

void rule_menu ()

generiert ein Untermenü: Rulemenu

void run (int *generationsToCalc*, int *ticksPerSecond*)

führt den Kern des Spiels aus

Parameters

generationsToCalc	
ticksPerSecond	

void settings_menu ()

generiert ein Untermenü: Settingsmenu

void start_game (int *is_random*)

diese Funktion startet ein Spiel

Parameters

is_random	legt fest ob das Spiel zufällig generiert sein soll

void start_menu ()

generiert ein Untermenü: startmenu

void tick (int * *end_game*, int * *pause_game*)

Diese Funktion berechnet die nächste Iteration des GOL.

Parameters

end_game	Verweis auf Int Value
pause_game	Verweis auf Int Value

game.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
#include "menu.h"
```

Data Structures

struct **cell**
struct **settings**

Functions

uint32_t **generate_random_int_msws** ()

Generiert eine Middle Square Weyl Sequence um daraus eine Randomzahl zu generieren. `https://pthree.org/2018/07/30/middle-square-weyl-sequence-prng/`
Diese Funktion wird verwendet, da so eine bessere Zufallszahlgenerierung gewährleistet wird.

void **alloc_grid** (struct **cell** ***grid_ptr, COORD gridsize)

*alloc_grid reserviert den Speicher des Feldes struct cell ***grid_ptr -> deklariere "grid" als pointer to pointer to pointer to struct cell*

void **dealloc_grid** (struct **cell** ***grid_ptr, const int x_size)

*dealloc_grid gibt den Speicher des Feldes wieder frei struct cell ***grid_ptr -> deklariere "grid" als pointer to pointer to pointer to struct cell*

void **copy_grid** (struct **cell** **grid_ptr_dest, struct **cell** **grid_ptr_src, COORD gridsize)

copy_grid kopiert alle Werte eines Feldes in ein neues Feld

void **load_preset_to_grid** (struct **cell** **grid_ptr, COORD gridsize)

load_preset_to_grid laed ein Preset aus einer Textdatei.

void **save_preset_from_grid** (struct **cell** **grid_ptr, COORD gridsize)

save_preset_to_grid speichert den Stand des Feldes als Preset in eine Textdatei.

int **count_living_neighbors** (struct **cell** grid)

count_living_neighbors zählt die lebenden umliegenden Nachbarn. diese Funktion ist möglichst effizient geschrieben, da sie beim initialisieren eines 1000x1000 Feldes bis zu 1.000.000x aufgerufen wird.

void **define_neighborhood** (struct **cell** **grid_ptr, COORD gridsize)

define_neighborhood definiert alle anliegenden Nachbarzellen für jede Zelle eines Feldes

void **add_neighborhood** (struct **cell** grid)

add_neighborhood informiert jede anliegende Nachbarzelle über den aktuellen Stand der Zelle das passiert durch das Hochzählen von livingNeighbors in jeder anliegenden Zelle diese Funktion ist möglichst effizient geschrieben, da sie bei einem 1000x1000 Feld bis zu 1.000.000x pro Tick aufgrufen werden kann.

void **sub_neighborhood** (struct **cell** grid)

sub_neighborhood informiert jede anliegende Nachbarzelle über den aktuellen Stand der Zelle das passiert durch das Runterzählen von livingNeighbors in jeder anliegenden Zelle diese Funktion ist möglichst effizient geschrieben, da sie bei einem 1000x1000 Feld bis zu 1.000.000x pro Tick aufgrufen werden kann.

void **initialize_empty_grid** (struct **cell** **grid_ptr, COORD gridsize)
initialize_empty_grid initialisiert ein leeres grid

void **calc_all_neighbors** (struct **cell** **grid_ptr, COORD gridsize)
calc_all_neighbors initialisiert die livingNeighbors für jede Zelle.

void **generate_random_grid** (struct **cell** **grid_ptr, COORD gridsize)
generate_random_grid initialisiert das übergebene Feld mit dem Modulo 2 (%2) aus einer zufälligen Zahl für jede Zelle eines Feldes

Function Documentation

void add_neighborhood (struct cell *grid*)

add_neighborhood informiert jede anliegende Nachbarzelle über den aktuellen Stand der Zelle das passiert durch das Hochzählen von livingNeighbors in jeder anliegenden Zelle diese Funktion ist möglichst effizient geschrieben, da sie bei einem 1000x1000 Feld bis zu 1.000.000x pro Tick aufgrufen werden kann.

Parameters

cell	Verweis auf eine Zelle im Spielfeld

void alloc_grid (struct cell *** *grid_ptr*, COORD *gridsize*)

alloc_grid reserviert den Speicher des Feldes struct cell ***grid_ptr -> deklariere "grid" als pointer to pointer to pointer to struct cell

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

void calc_all_neighbors (struct cell ** *grid_ptr*, COORD *gridsize*)

calc_all_neighbors initialisiert die livingNeighbors für jede Zelle.

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

void copy_grid (struct cell ** *grid_ptr_dest*, struct cell ** *grid_ptr_src*, COORD *gridsize*)

copy_grid kopiert alle Werte eines Feldes in ein neues Feld

Parameters

grid_ptr_dest	Verweis auf das Ziel Grid
grid_ptr_src	Verweis auf das Grid von dem kopiert werden soll
gridsize	Größe des Spielfeldes

int count_living_neighbors (struct cell *grid*)

count_living_neighbors zählt die lebenden umliegenden Nachbarn. diese Funktion ist möglichst effizient geschrieben, da sie beim initialisieren eines 1000x1000 Feldes bis zu 1.000.000x aufgerufen wird.

Parameters

cell	Verweis auf eine Zelle im Spielfeld

Returns

int Anzahl der lebenden Nachbarn

void dealloc_grid (struct cell *** *grid_ptr*, const int *x_size*)

dealloc_grid gibt den Speicher des Feldes wieder frei struct cell ***grid_ptr -> deklariere "grid" als pointer to pointer to pointer to struct cell

Parameters

grid_ptr	Verweis auf das Grid
x_size	Größe der X-Achse des Spielfeldes

void define_neighborhood (struct cell ** *grid_ptr*, COORD *gridsize*)

define_neighborhood definiert alle anliegenden Nachbarzellen für jede Zelle eines Feldes

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

void generate_random_grid (struct cell ** *grid_ptr*, COORD *gridsize*)

generate_random_grid initialisiert das übergebene Feld mit dem Modulo 2 (%2) aus einer zufälligen Zahl für jede Zelle eines Feldes

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

uint32_t generate_random_int_msws ()

Generiert eine Middle Square Weyl Sequence um daraus eine Randomzahl zu generieren.

Returns

uint32_t gibt einen unsigned 32bit Integer zurück

void initialize_empty_grid (struct cell ** *grid_ptr*, COORD *gridsize*)

initialize_empty_grid initialisiert ein leeres grid

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

void load_preset_to_grid (struct cell ** *grid_ptr*, COORD *gridsize*)

load_preset_to_grid laed ein Preset aus einer Textdatei.

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

void save_preset_from_grid (struct cell ** *grid_ptr*, COORD *gridsize*)

save_preset_to_grid speichert den Stand des Feldes als Preset in eine Textdatei.

Parameters

grid_ptr	Verweis auf das Grid
gridsize	Größe des Spielfeldes

void sub_neighborhood (struct cell *grid*)

sub_neighborhood informiert jede anliegende Nachbarzelle über den aktuellen Stand der Zelle das passiert durch das Runterzählen von livingNeighbors in jeder anliegenden Zelle diese Funktion ist möglichst effizient geschrieben, da sie bei einem 1000x1000 Feld bis zu 1.000.000x pro Tick aufgrufen werden kann.

Parameters

cell	Verweis auf eine Zelle im Spielfeld

menu.h File Reference

```
#include <windows.h>
#include <string.h>
#include "game.h"
```

Data Structures

struct **menu_button**

Functions

void **print_logo** (int x, int y)
gibt das Spiel-Logo an Position x,y aus

void **set_cursor** (int x, int y)
Setzt den Consolecursor an eine Position im Consolenbuffer.

void **set_fontsize** (int size)
Setzt die Consolenfontsize.

void **draw_menu** (struct **menu_button** Menu_Button[3], int array_length)
Zeichnet ein Menu anhand von Buttons.

void **draw_settings_menu_values** (struct **menu_button** Menu_Button[3], int array_length, struct **settings** gamesettings)
Zeichnet die Werte im Settingsmenü

void **draw_rule_menu_values** (struct **menu_button** Menu_Button[3], int array_length, struct rule gamerules)
Zeichnet die Werte im Rulemenü

void **edit_setting_value** (struct **settings** *gamesettings, struct **menu_button** Menu_Button[5], int cursor_pos)
Initialisiert eine Scanf-Sequenz im Settingesmenü um einen Wert anzupassen.

void **edit_rule_value** (struct rule *gamerules, struct **menu_button** Menu_Button[5], int cursor_pos)
Initialisiert eine Scanf-Sequenz im Rulesmenü um einen Wert anzupassen.

void **set_value_cursor** (struct **menu_button** Menu_Button[5], int cursor_pos)
Bewegt den Cursor im Settingsmenü an eine übergebene Stelle.

void **set_menucursor** (struct **menu_button** Menu_Button[3], int array_length, int position)
Bewegt den Cursor im Mainmenü an eine übergebene Stelle.

void **draw_cursor** (COORD cords)
Zeichnet den Menücursor an einer bestimmte Stelle.

void **erase_cursor** (COORD cords)
Löscht den Menücursor an einer bestimmte Stelle.

COORD **get_console_window_size** (HANDLE hConsoleOutput)
Get the console window size object.

void **set_console_fullscreen** ()
Setzt die Console auf Vollbild (so wie alt+enter)

Function Documentation

void draw_cursor (COORD *cords*)

Zeichnet den Menücursor an einer bestimmte Stelle.

Parameters

cords	Coords der Position

void draw_menu (struct menu_button *Menu_Button*[3], int *array_length*)

Zeichnet ein Menu anhand von Buttons.

Parameters

Menu_Button	Array mit Menubuttons
array_length	Arraylänge

void draw_rule_menu_values (struct menu_button *Menu_Button*[3], int *array_length*, struct rule *gamerules*)

Zeichnet die Werte im Rulemenü

Parameters

Menu_Button	Array mit Menubuttons
array_length	Arraylänge
gamerules	Gamerule Struct

void draw_settings_menu_values (struct menu_button *Menu_Button*[3], int *array_length*, struct settings *gamesettings*)

Zeichnet die Werte im Settingsmenü

Parameters

Menu_Button	Array mit Menubuttons
array_length	Arraylänge
gamesettings	Gamesettings Struct

void edit_rule_value (struct rule * *gamerules*, struct menu_button *Menu_Button*[5], int *cursor_pos*)

Initialisiert eine Scanf-Sequenz im Rulesmenü um einen Wert anzupassen.

Parameters

gamerules	Gamesrules Struct
Menu_Button	Array mit Menubuttons
cursor_pos	Position des Cursors

void edit_setting_value (struct settings * *gamesettings*, struct menu_button *Menu_Button*[5], int *cursor_pos*)

Initialisiert eine Scanf-Sequenz im Settingesmenü um einen Wert anzupassen.

Parameters

gamesettings	Gamesettings Struct
Menu_Button	Array mit Menubuttons
cursor_pos	Position des Cursors

void erase_cursor (COORD *cords*)

Löscht den Menücursor an einer bestimmte Stelle.

Parameters

cords	Coords der Position

COORD get_console_window_size (HANDLE *hConsoleOutput*)

Get the console window size object.

Parameters

hConsoleOutput	Console Handle

Returns

COORD Größe der Console als Coord-Objekt

void print_logo (int *x*, int *y*)

gibt das Spiel-Logo an Position x,y aus

Parameters

x	X-Position im Consolenbuffer
y	Y-Position im Consolenbuffer

void set_console_fullscreen ()

Setzt die Console auf Vollbild (so wie alt+enter)

void set_cursor (int *x*, int *y*)

Setzt den Consolecursor an eine Position im Consolenbuffer.

Parameters

x	X-Position im Consolenbuffer
y	Y-Position im Consolenbuffer

void set_fontsize (int *size*)

Setzt die Consolenfontsize.

Parameters

size	Schriftgröße

void set_menucursor (struct menu_button *Menu_Button*[3], int *array_length*, int *position*)

Bewegt den Cursor im Mainmenü an eine übergebene Stelle.

Parameters

Menu_Button	Array mit Menubuttons
array_length	Arraylänge
position	Position des Cursors

void set_value_cursor (struct menu_button *Menu_Button*[5], int *cursor_pos*)

Bewegt den Cursor im Settingsmenü an eine übergebene Stelle.

Parameters

Menu_Button	Array mit Menubuttons
cursor_pos	Position des Cursors

buffer.h File Reference

```
#include <stdint.h>
#include <stdlib.h>
#include "menu.h"
#include "game.h"
```

Functions

void **alloc_buffer** (char **buffer, COORD gridsize)
aloziiert den Buffer

void **dealloc_buffer** (char **buffer)
dealoziiert den Buffer

void **initialize_buffer** (char *buffer, COORD gridsize, char symbolAlive, char symbolDead)
initialisiert den Buffer

void **revive_buffer_at_coord** (char *buffer, struct **settings** gamesettings, int x_pos, int y_pos)
Diese Funktion belebt eine Zelle an bestimmter Stelle im Buffer wieder.

void **kill_buffer_at_coord** (char *buffer, struct **settings** gamesettings, int x_pos, int y_pos)
Diese Funktion tötet eine Zelle an bestimmter Stelle im Buffer.

int **calc_buffer_size** (COORD gridsize)
Berechnet die Speichergröße des Buffers.

int **calc_position_in_buffer** (COORD gridsize, const int x_pos, const int y_pos)
Berechnet die Position einer Zelle im Buffer anhand der Position auf dem Spielfeld.

void **draw_hud** (struct **settings** gamesettings, int aliveCells, int currentGeneration, int generationsToCalc)
Zeichnet das Head-Up-Display im Spiel.

void **print_buffer** (char *buffer)
Gibt den Buffer aus.

Function Documentation

void alloc_buffer (char ** *buffer*, COORD *gridsize*)

aloziiert den Buffer

Parameters

buffer	Verweis auf den Buffer
gridsize	Größe des Spielfeldes

int calc_buffer_size (COORD *gridsize*)

Berechnet die Speichergröße des Buffers.

Parameters

gridsize	Größe des Spielfeldes

Returns

int Gibt die Speichergröße zurück

int calc_position_in_buffer (COORD *gridsize*, const int *x_pos*, const int *y_pos*)

Berechnet die Position einer Zelle im Buffer anhand der Position auf dem Spielfeld.

Parameters

gridsize	Größe des Spielfeldes
x_pos	Position X auf dem Spielfeld
y_pos	Position Y auf dem Spielfeld

Returns

int Gibt die Position im Buffer zurück

void dealloc_buffer (char ** *buffer*)

dealoziiert den Buffer

Parameters

buffer	Verweis auf den Buffer

void draw_hud (struct settings *gamesettings*, int *aliveCells*, int *currentGeneration*, int *generationsToCalc*)

Zeichnet das Head-Up-Display im Spiel.

Parameters

gamesettings	Größe des Spielfeldes
aliveCells	Anzahl lebender Zellen
currentGeneration	Nummer der aktuellen Generation
generationsToCalc	Noch zu berechnende Generationen

void initialize_buffer (char * *buffer*, COORD *gridsize*, char *symbolAlive*, char *symbolDead*)

initialisiert den Buffer

Parameters

buffer	Verweis auf den Buffer
gridsize	Größe des Spielfeldes
symbolAlive	Symbol für lebende Zellen
symbolDead	Symbol für tote Zellen

void kill_buffer_at_coord (char * *buffer*, struct settings *gamesettings*, int *x_pos*, int *y_pos*)

Diese Funktion tötet eine Zelle an bestimmter Stelle im Buffer.

Parameters

buffer	Verweis auf den Buffer
gamesettings	Gamesettings Struct
x_pos	Position X auf dem Spielfeld
y_pos	Position Y auf dem Spielfeld

void print_buffer (char * *buffer*)

Gibt den Buffer aus.

Parameters

buffer	Verweis auf den Buffer

void revive_buffer_at_coord (char * *buffer*, struct settings *gamesettings*, int *x_pos*, int *y_pos*)

Diese Funktion belebt eine Zelle an bestimmter Stelle im Buffer wieder.

Parameters

buffer	Verweis auf den Buffer
gamesettings	Gamesettings Struct
x_pos	Position X auf dem Spielfeld
y_pos	Position Y auf dem Spielfeld