Game of Life

Generated by Doxygen 1.9.8

Chapter 1

Game of Life by David Zoller

1.1 Introduction

This is the introduction.

1.2 Installation

1.2.1 Step 1: Opening the box

etc...

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 Color_theme Struct Reference

Represents a color theme with 5 colors.

`#include <Color.h>`

Data Fields

- SDL_Color prim
- SDL_Color primacc
- SDL_Color sec
- SDL_Color secacc
- SDL_Color bg

4.1.1 Detailed Description

Represents a color theme with 5 colors.

4.1.2 Field Documentation

4.1.2.1 bg

`SDL_Color Color_theme::bg`

Background color

4.1.2.2 prim

`SDL_Color Color_theme::prim`

Primary color

4.1.2.3 primacc

`SDL_Color Color_theme::primacc`

Primary accent color

4.1.2.4 sec

`SDL_Color Color_theme::sec`

Secondary color

4.1.2.5 secacc

`SDL_Color Color_theme::secacc`

Secondary accent color

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

- Color.h

4.2 Fgame_file Struct Reference

Data Fields

- char * **path**
- SDL_Rect **location**

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

- File.h

4.3 gameArea Struct Reference

Játéktér és tulajdonságai.

`#include <gameArea.h>`

Data Fields

- size_t **w**
- size_t **h**
- uint8_t ** **area**
- uint8_t **history_lenght**

4.3.1 Detailed Description

Játéktér és tulajdonságai.

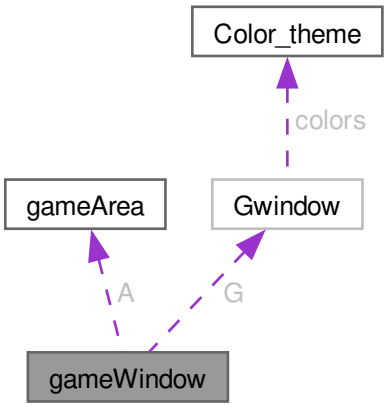
Létrehozni csak fügvénnyel szabad, törlése kötelező az Afree fügvénnyel

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

- gameArea.h

4.4 gameWindow Struct Reference

Collaboration diagram for gameWindow:



Data Fields

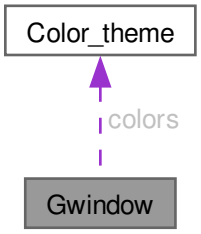
- gameArea **A**
- Gwindow **G**
- char * **name**
- SDL_Texture * **pre_rendered_cells**
- SDL_Texture * **full_game**
- size_t **texture_w**
- size_t **texture_h**
- double **zoom**
- ssize_t **x_screen_offset**
- ssize_t **y_screen_offset**
- SDL_TimerID **autoplay_id**
- Uint32 **autoplay_delay**

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

- gameWindow.h

4.5 Gwindow Struct Reference

Collaboration diagram for Gwindow:



Data Fields

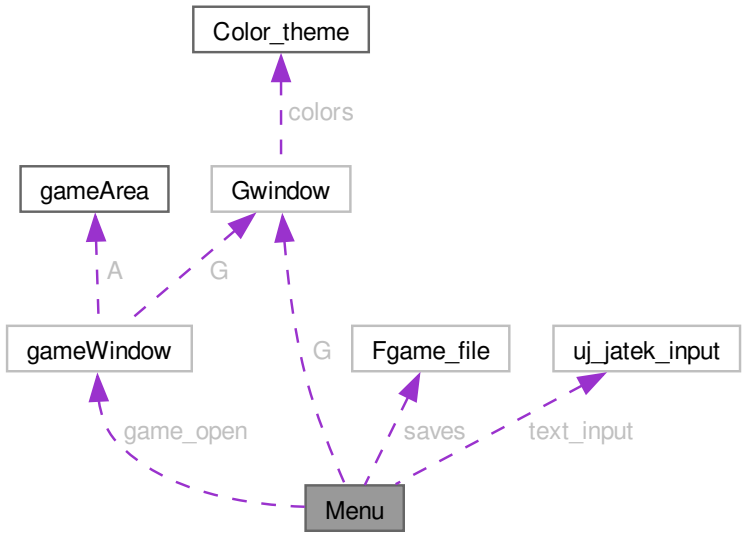
- SDL_Window ∗ **win**
- SDL_Renderer ∗ **ren**
- size_t **w**
- size_t **h**
- TTF_Font ∗ **font_big**
- TTF_Font ∗ **font_reg**
- Color_theme **colors**

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

- Graphics.h

4.6 Menu Struct Reference

Collaboration diagram for Menu:



Data Fields

- Gwindow **G**
- Fgame_file * **saves**
- size_t **save_cnt**
- gameWindow **game_open**
- uj_jatek_input **text_input**

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

- Menu.h

4.7 uj_jatek_input Struct Reference

Data Fields

- char **name** [INPUT_MAX_LENGHT+4]
- SDL_Rect **name_rct**
- char **width** [INPUT_MAX_LENGHT]
- SDL_Rect **width_rct**
- char **height** [INPUT_MAX_LENGHT]
- SDL_Rect **height_rct**
- SDL_Rect **button**

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

- Menu.h

Chapter 5

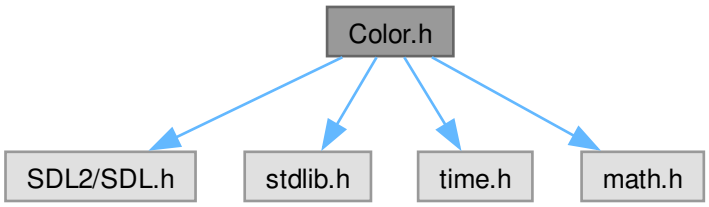
File Documentation

5.1 Color.h File Reference

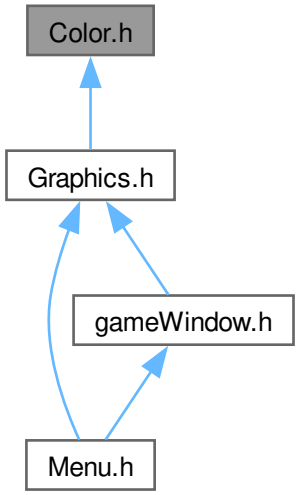
Header file for color-related structures and functions.

```
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
```

Include dependency graph for Color.h:



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



Data Structures

- struct Color_theme

  Represents a color theme with 5 colors.

Typedefs

- typedef enum Colortype **Colortype**
- typedef struct Color_theme **Color_theme**

Enumerations

- enum Colortype { **primary** , **secondary** , **primary_accent** , **secondary_accent** }

  Represents different color types for rendering.

Functions

- Color_theme Cinit ()

  Initializes a Color_theme with a dynamically generated color scheme.

5.1.1 Detailed Description

Header file for color-related structures and functions.

5.1.2 Function Documentation

5.1.2.1 Cinit()

```
Color_theme Cinit ( )
```

Initializes a Color_theme with a dynamically generated color scheme.

This function initializes a Color_theme structure with dynamically generated colors based on a random hue value that has a higher probability to be a warm color. From this hue, a complement color is generated for the secondary. The function ensures that if called multiple times, it returns the same theme.

Returns

A Color_theme structure representing the generated color scheme.

5.2 Color.h

Go to the documentation of this file.

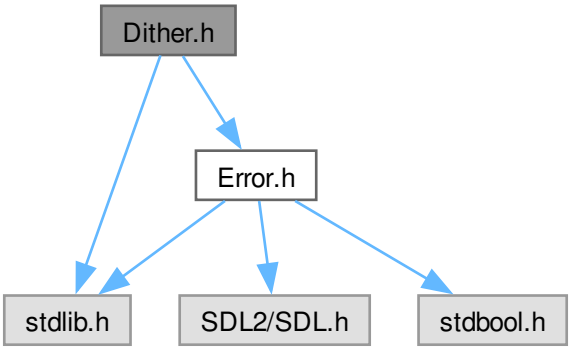
```
00001 #ifndef COLOR_H
00002 #define COLOR_H
00003
00004 #include <SDL2/SDL.h>
00005 #include <stdlib.h>
00006 #include <time.h>
00007 #include <math.h>
00008
00018 typedef enum Colortype
00019 {
00020     primary,
00021     secondary,
00022     primary_accent,
00023     secondary_accent
00024 } Colortype;
00025
00030 typedef struct Color_theme {
00031     SDL_Color prim;
00032     SDL_Color primacc;
00033     SDL_Color sec;
00034     SDL_Color secacc;
00035     SDL_Color bg;
00036 } Color_theme;
00037
00038
00049 Color_theme Cinit();
00050 #endif
```

5.3 Dither.h File Reference

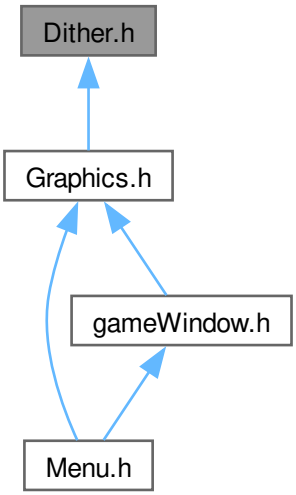
File for functions related to ordered dithering used for cell fadeout.

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

Include dependency graph for Dither.h:



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



Functions

- size_t ** Dgenerate_bayer_matrix (size_t n)

  Generates a Bayer matrix for ordered dithering.
- void Dfree_bayer_matrix (size_t **matrix)

  Deallocates memory used by a Bayer matrix.

5.3.1 Detailed Description

File for functions related to ordered dithering used for cell fadeout.

Refer to https://en.wikipedia.org/wiki/Ordered_dithering for the equation source.

5.3.2 Function Documentation

5.3.2.1 Dfree_bayer_matrix()

```
void Dfree_bayer_matrix (
            size_t ** matrix )
```

Deallocates memory used by a Bayer matrix.

Parameters

matrix	The Bayer matrix to be freed.

5.3.2.2 Dgenerate_bayer_matrix()

```
size_t ** Dgenerate_bayer_matrix (
            size_t n )
```

Generates a Bayer matrix for ordered dithering.

Parameters

n	The side length of the matrix.

Returns

A new Bayer matrix. Memory deallocation with Dfree_bayer_matrix is the caller's responsibility.

5.4 Dither.h

Go to the documentation of this file.

```
00001 #ifndef DITHER_H
00002 #define DITHER_H
00003
00004 #include <stdlib.h>
00005 #include "Error.h"
00006
00019 size_t **Dgenerate_bayer_matrix(size_t n);
00020
00026 void Dfree_bayer_matrix(size_t **matrix);
00027 #endif
```

5.5 Error.h

```
00001 #ifndef ERROR_H
00002 #define ERROR_H
00003
00004 #include <SDL2/SDL.h>
00005 #include <stdbool.h>
00006 #include <stdlib.h>
00007
00008 #define ErrorIFtrue(test, error_msg) ErrorIFtrue_with_params(test, error_msg, __FILE__, __LINE__);
00009 #define ErrorIFnull(ptr, error_msg) ErrorIFtrue_with_params(ptr == NULL, error_msg, __FILE__,
      __LINE__);
00010 #define ErrorIFsdl(func_with_negative_error) ErrorIFtrue_with_params(func_with_negative_error < 0,
      "SDL hiba!", __FILE__, __LINE__);
00011 #define ErrorIFnoMemory(ptr) ErrorIFtrue_with_params(ptr == NULL, "Nincs eleg memoria!", __FILE__,
      __LINE__);
00012
00013 void ErrorIFtrue_with_params(bool test, char* error_msg, char* FILE, int LINE);
00014
00015 #endif
```

5.6 File.h

```
00001 #ifndef FILE_H
00002 #define FILE_H
00003
00004 #include <stdlib.h>
00005 #include <dirent.h>
00006 #include <string.h>
00007
00008 #include "Error.h"
00009 #include "gameArea.h"
00010 #include "debugmalloc.h"
00011
00012 #define SAVES_FOLDER "saved/"
00013
00014 typedef struct Fgame_file{
00015     char *path;
00016     SDL_Rect location;
00017 } Fgame_file;
00018
00019 gameArea Fopen(char *path);
00020 void Fsave(char *path, gameArea *gamearea);
00021 size_t Flist(Fgame_file games[], size_t max_count);
00022
00023 #endif
```

5.7 gameArea.h

```
00001 #ifndef GAMEAREA_H
00002 #define GAMEAREA_H
00003
00004 #include <SDL2/SDL.h>
00005 #include <stdlib.h>
00006 #include <stdbool.h>
00007
00008 #include "Error.h"
00009 #include "debugmalloc.h"
00010
00011
00017 typedef struct gameArea {
00018     size_t w;       // A játéktér szélessége
00019     size_t h;       // A játéktér magassága
00020     uint8_t **area;  // A játéktér tömbje
00021     uint8_t history_lenght;
00022 } gameArea;
00023
00024 gameArea Anew(size_t width, size_t height);
00025 void Aclear(gameArea *gamearea);
00026 void Afree(gameArea *gamearea);
00027 ssize_t Agetage(uint8_t cell);
00028 void Astep(gameArea *A);
00029 bool Aback(gameArea *A);
00030 void Aflipcell(gameArea *A, double x, double y);
00031
00032 #endif
```

5.8 gameWindow.h

```
00001 #ifndef GAMEWINDOW_H
00002 #define GAMEWINDOW_H
00003
00004 #include <stdbool.h>
00005 #include "Graphics.h"
00006 #include "gameArea.h"
00007 #include "File.h"
00008
00009 typedef struct gameWindow {
00010     gameArea A;
00011     Gwindow G;
00012     char *name;
00013     SDL_Texture *pre_rendered_cells;
00014     SDL_Texture *full_game;
00015     size_t texture_w;
00016     size_t texture_h;
00017     double zoom;
00018     ssize_t x_screen_offset;
00019     ssize_t y_screen_offset;
00020     SDL_TimerID autoplay_id;
00021     Uint32 autoplay_delay;
00022 } gameWindow;
00023
00024 gameWindow Winit(gameArea A, char *name);
00025 void Wclose(gameWindow *game);
00026 void Wclick(gameWindow *game, int x, int y);
00027 void Wdraw(gameWindow *game, bool all_cells);
00028 void Wzoom(gameWindow *game, double wheel, int x, int y);
00029 void Wresetzoom(gameWindow *game);
00030 void Wevent(gameWindow *game, SDL_Event *e);
00031 #endif
```

5.9 Graphics.h

```
00001 #ifndef GRAPHICS_H
00002 #define GRAPHICS_H
00003
00004 #include <SDL2/SDL.h>
00005 #include <SDL2/SDL2_gfxPrimitives.h>
00006 #include <SDL2/SDL_ttf.h>
00007 #include <stdbool.h>
00008
00009 #include "Color.h"
00010 #include "Dither.h"
00011 #include "Error.h"
00012
00013 #define CELL_SIZE 8
00014
00015 typedef struct Gwindow {
00016     SDL_Window *win;
00017     SDL_Renderer *ren;
00018     size_t w, h;
00019     TTF_Font *font_big;
00020     TTF_Font *font_reg;
00021     Color_theme colors;
00022 } Gwindow;
00023
00024 void Ginit();
00025 void Gclose(Gwindow *window);
00026 void Gquit();
00027 Gwindow Gnew(char title[], int width, int height, bool resizable);
00028 void Gset_color(Gwindow *window, SDL_Color col);
00029 void Gfill_background(Gwindow *window);
00030 void Gprint_title(Gwindow *window);
00031 SDL_Rect Gprint(Gwindow *window, char *text, SDL_Rect *location, Colortype col);
00032 void Gtextbox(Gwindow *window, char *text, SDL_Rect *location, Colortype col, size_t border_width);
00033 SDL_Texture *Gpre_render_cells(Gwindow *window);
00034 void Ginput_text(Gwindow *window, char *dest, size_t hossz, SDL_Rect teglalap, bool is_file_name);
00035 #endif
```

5.10 Menu.h

```
00001 #ifndef MENU_H
00002 #define MENU_H
00003
00004
00005 #include <stdlib.h>
```

```
00006
00007 #include "Graphics.h"
00008 #include "File.h"
00009 #include "gameWindow.h"
00010 #include "debugmalloc.h"
00011
00012 #define MAX_SAVES 13
00013 #define INPUT_MAX_LENGHT 15
00014
00015 typedef struct uj_jatek_input{
00016     char name[INPUT_MAX_LENGHT+4];
00017     SDL_Rect name_rct;
00018     char width[INPUT_MAX_LENGHT];
00019     SDL_Rect width_rct;
00020     char height[INPUT_MAX_LENGHT];
00021     SDL_Rect height_rct;
00022     SDL_Rect button;
00023 } uj_jatek_input;
00024
00025 typedef struct Menu{
00026     Gwindow G;
00027     Fgame_file *saves;
00028     size_t save_cnt;
00029     gameWindow game_open;
00030     uj_jatek_input text_input;
00031 } Menu;
00032
00033 Menu Minit();
00034 void Mclose(Menu *menu);
00035 void Mclick(Menu *menu, int x, int y);
00036 void Mevent(Menu *menu, SDL_Event *e);
00037
00038 #endif
```

Index