

Game of Life

Generated by Doxygen 1.9.8

1 Game of Life by David Zoller	1
1.1 Introduction	1
1.1.1 Conway's Game of Life	1
1.1.2 Rules	1
1.2 Structure	2
1.3 File Format (.con)	2
1.4 Installation	2
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 Color_theme Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 bg	7
4.1.2.2 prim	7
4.1.2.3 primacc	8
4.1.2.4 sec	8
4.1.2.5 secacc	8
4.2 Fgame_file Struct Reference	8
4.2.1 Detailed Description	8
4.3 gameArea Struct Reference	8
4.3.1 Detailed Description	9
4.4 gameWindow Struct Reference	9
4.4.1 Detailed Description	10
4.5 Gwindow Struct Reference	10
4.5.1 Detailed Description	11
4.6 Menu Struct Reference	11
4.6.1 Detailed Description	12
4.7 uj_jatek_input Struct Reference	12
4.7.1 Detailed Description	12
5 File Documentation	13
5.1 Color.h File Reference	13
5.1.1 Detailed Description	14
5.1.2 Enumeration Type Documentation	14
5.1.2.1 Colortype	14
5.1.3 Function Documentation	15
5.1.3.1 Cinit()	15
5.2 Color.h	15

5.3 Dither.h File Reference	16
5.3.1 Detailed Description	17
5.3.2 Function Documentation	17
5.3.2.1 Dfree_bayer_matrix()	17
5.3.2.2 Dgenerate_bayer_matrix()	17
5.4 Dither.h	18
5.5 Error.h File Reference	18
5.5.1 Detailed Description	19
5.5.2 Macro Definition Documentation	20
5.5.2.1 ErrorIFnoMemory	20
5.5.2.2 ErrorIFnull	20
5.5.2.3 ErrorIFsdl	20
5.5.2.4 ErrorIFtrue	20
5.5.3 Function Documentation	21
5.5.3.1 ErrorIFtrue_with_params()	21
5.5.3.2 ErrorWarning()	21
5.6 Error.h	21
5.7 File.h File Reference	22
5.7.1 Detailed Description	23
5.7.2 Function Documentation	24
5.7.2.1 Flist()	24
5.7.2.2 Fopen()	24
5.7.2.3 Fsave()	24
5.8 File.h	25
5.9 gameArea.h File Reference	25
5.9.1 Detailed Description	27
5.9.2 Function Documentation	27
5.9.2.1 Aback()	27
5.9.2.2 Aclear()	27
5.9.2.3 Aflipcell()	27
5.9.2.4 Afree()	28
5.9.2.5 Agetage()	28
5.9.2.6 Anew()	28
5.9.2.7 Astep()	29
5.10 gameArea.h	29
5.11 gameWindow.h File Reference	29
5.11.1 Detailed Description	31
5.11.2 Function Documentation	31
5.11.2.1 Wclick()	31
5.11.2.2 Wclose()	31
5.11.2.3 Wdraw()	31
5.11.2.4 Wevent()	32

5.11.2.5 Winit()	32
5.11.2.6 Wresetzoom()	32
5.11.2.7 Wzoom()	32
5.12 gameWindow.h	33
5.13 Graphics.h File Reference	33
5.13.1 Detailed Description	35
5.13.2 Macro Definition Documentation	35
5.13.2.1 CELL_SIZE	35
5.13.3 Function Documentation	35
5.13.3.1 Gclose()	35
5.13.3.2 Gfill_background()	36
5.13.3.3 Ginit()	36
5.13.3.4 Ginput_text()	36
5.13.3.5 Gnew()	37
5.13.3.6 Gprint()	37
5.13.3.7 Gprint_title()	37
5.13.3.8 Gquit()	38
5.13.3.9 Gset_color()	38
5.13.3.10 Gtextbox()	38
5.14 Graphics.h	39
5.15 Menu.h File Reference	39
5.15.1 Detailed Description	40
5.15.2 Function Documentation	41
5.15.2.1 Mclick()	41
5.15.2.2 Mclose()	42
5.15.2.3 Mevent()	42
5.15.2.4 Minit()	42
5.16 Menu.h	42
Index	45

Chapter 1

Game of Life by David Zoller

1.1 Introduction

This is a Game of Life simulator implemented in C using the SDL2 library for graphics. It is made as a project for the Programming 1 course at the Budapest University of Technology and Economics.

The simulator allows users to create, save, and load different game states, and control the simulation with play, pause, step forward, and step back functions.

1.1.1 Conway's Game of Life

The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. It is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves. It is Turing complete and can simulate a universal constructor or any other Turing machine.

1.1.2 Rules

The universe of the Game of Life is an infinite (this simulator only works on finite grids), two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead (or populated and unpopulated, respectively). Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if by underpopulation.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any live cell with more than three live neighbours dies, as if by overpopulation.
- Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

The generations are created by applying the above rules simultaneously to every cell in the seed, live or dead; births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a tick. Each generation is a pure function of the preceding one.

See also

Source:

Wikipedia https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

1.2 Structure

The project is structured into several modules:

- Graphics: Handles all the graphical output using the SDL2 library.
- [Menu](#): Handles the main menu where users can create, load, and save games.
- [gameWindow](#): Handles the window where the simulation is displayed and controlled.
- [gameArea](#): Represents the game area where the cells live.
- File: Handles file operations for saving and loading game states.
- Color: Defines the color theme used in the graphics.
- Dither: Provides functions for dithering colors.
- Error: Provides functions for error handling.

The state of the cells are stored in a 2D array of 8bit unsigned integers, where the LSB represents the current state of the cell, and the other 7 bits represent the history of the cell. This allows for the simulation to be run in reverse for 7 ticks and to show the decay of the cells.

For example:

```
00000000 - Dead cell
00000010 - Dead cell, was alive 1 tick ago
01010001 - Alive cell, was alive 4 and 6 ticks ago
```

1.3 File Format (.con)

The first line of the file specifies the dimensions of the game board, separated by spaces (width first, then height). Additional data can be stored within the same line. The game board follows next. Here, empty cells are represented by a dot, while living cells are represented by a capital 'O'. These are stored in a grid layout. The file type is ".con".

For example: elso.con

```
7 3
.....O.
OOO..O.
.....O.
```

1.4 Installation

The project uses the `SDL2` and `gcc` libraries, which need to be installed before building the project. The project can be built using the `build.sh` script. The project can be run by running the main `GameOfLife` executable. They should be run from directory where they are located.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Color_theme	A structure representing a color theme with five colors	7
Fgame_file	Structure representing a game file	8
gameArea	Represents the game area and its properties	8
gameWindow	Represents the game window and its properties	9
Gwindow	Represents the graphics window and its properties	10
Menu	Represents the menu and its properties	11
uj_jatek_input	Represents the input for a new game	12

Chapter 3

File Index

3.1 File List

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

Color.h	This file contains color-related structures and functions	13
Dither.h	This file contains functions for generating and deallocating a Bayer matrix for ordered dithering	16
Error.h	This file contains error handling macros and functions	18
File.h	File operations for the game	22
gameArea.h	This file contains the structures and functions for the gameArea , where the simulation takes place and the cells live	25
gameWindow.h	This file contains the structures and functions for the game window	29
Graphics.h	This file contains code used for graphics	33
Menu.h	This file contains the structures and functions for the menu window	39

Chapter 4

Data Structure Documentation

4.1 Color_theme Struct Reference

A structure representing a color theme with five 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

A structure representing a color theme with five colors.

4.1.2 Field Documentation

4.1.2.1 bg

```
SDL_Color Color_theme::bg
```

Background color of the theme

4.1.2.2 prim

```
SDL_Color Color_theme::prim
```

Primary color of the theme

4.1.2.3 primacc

```
SDL_Color Color_theme::primacc
```

Primary accent color of the theme

4.1.2.4 sec

```
SDL_Color Color_theme::sec
```

Secondary color of the theme

4.1.2.5 secacc

```
SDL_Color Color_theme::secacc
```

Secondary accent color of the theme

4.2 Fgame_file Struct Reference

Structure representing a game file.

```
#include <File.h>
```

Data Fields

- **char * path**
Path to the game file.
- **SDL_Rect location**
Location of the opening button on screen.

4.2.1 Detailed Description

Structure representing a game file.

4.3 gameArea Struct Reference

Represents the game area and its properties.

```
#include <gameArea.h>
```

Data Fields

- **size_t w**
Width of the game area.
- **size_t h**
Height of the game area.
- **uint8_t ** area**
Array representing the game area, least significant bit is the current state, from that the next 7 bits are the history of the cell.
- **uint8_t history_length**
History length of the game area, maximum 7.

4.3.1 Detailed Description

Represents the game area and its properties.

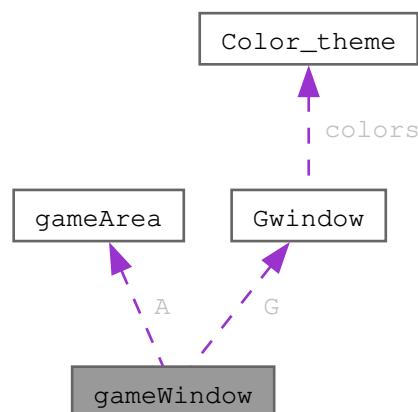
This structure should only be created with a function, and must be deleted with the Afree function.

4.4 gameWindow Struct Reference

Represents the game window and its properties.

```
#include <gameWindow.h>
```

Collaboration diagram for gameWindow:



Data Fields

- [gameArea](#) **A**
The game area.
- [Gwindow](#) **G**
The graphics window.
- char * **name**
The name of the game window.
- SDL_Texture * **pre_rendered_cells**
The pre-rendered cells.
- double **zoom**
The zoom level.
- ssize_t **x_screen_offset**
The x-coordinate screen offset.
- ssize_t **y_screen_offset**
The y-coordinate screen offset.
- SDL_TimerID **autoplay_id**
The autoplay timer ID.
- Uint32 **autoplay_delay**
The autoplay delay.

4.4.1 Detailed Description

Represents the game window and its properties.

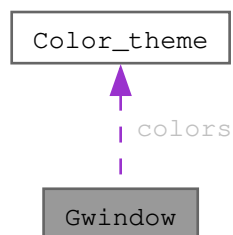
This structure should only be created with a function, and must be deleted with the Wclose function.

4.5 Gwindow Struct Reference

Represents the graphics window and its properties.

```
#include <Graphics.h>
```

Collaboration diagram for Gwindow:



Data Fields

- `SDL_Window * win`
The SDL window.
- `SDL_Renderer * ren`
The SDL renderer.
- `size_t w`
- `size_t h`
The width and height of the window.
- `TTF_Font * font_big`
The font used for the title.
- `TTF_Font * font_reg`
The font used for regular text.
- `Color_theme colors`
The color theme.

4.5.1 Detailed Description

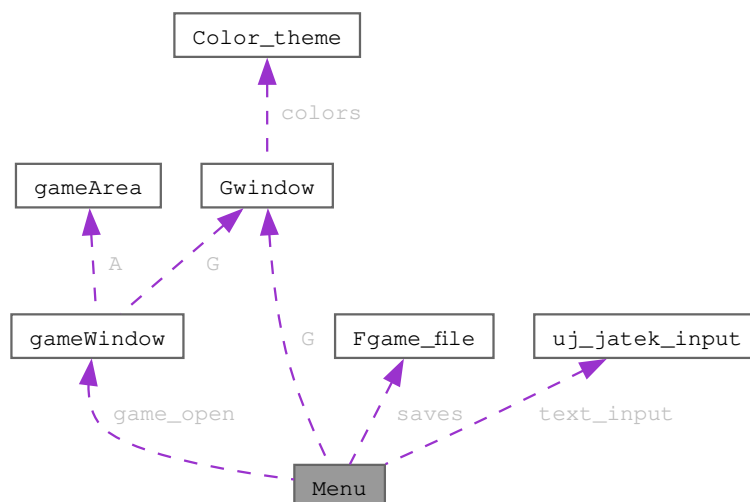
Represents the graphics window and its properties.

4.6 Menu Struct Reference

Represents the menu and its properties.

```
#include <Menu.h>
```

Collaboration diagram for Menu:



Data Fields

- [Gwindow](#) **G**
The graphics window.
- [Fgame_file](#) * **saves**
The saved games.
- `size_t` **save_cnt**
The count of saved games.
- [gameWindow](#) **game_open**
The open game window.
- [uj_jatek_input](#) **text_input**
The input for a new game.

4.6.1 Detailed Description

Represents the menu and its properties.

This structure should only be created with Minit, and must be deleted with the Mclose function.

4.7 uj_jatek_input Struct Reference

Represents the input for a new game.

```
#include <Menu.h>
```

Data Fields

- `char` **name** [INPUT_MAX LENGHT+4]
The name of the new game.
- `SDL_Rect` **name_rct**
The text box for the name input.
- `char` **width** [INPUT_MAX LENGHT]
The width of the new game.
- `SDL_Rect` **width_rct**
The text box for the width input.
- `char` **height** [INPUT_MAX LENGHT]
The height of the new game.
- `SDL_Rect` **height_rct**
The text box for the height input.
- `SDL_Rect` **button**
The bounding box for the new game button.

4.7.1 Detailed Description

Represents the input for a new game.

Chapter 5

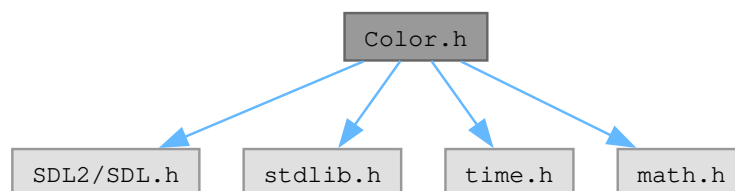
File Documentation

5.1 Color.h File Reference

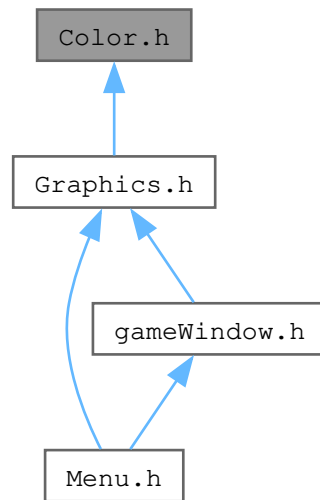
This file contains color-related structures and functions.

```
#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](#)
A structure representing a color theme with five colors.

Typedefs

- typedef enum [Colortype](#) **Colortype**
- typedef struct [Color_theme](#) **Color_theme**

Enumerations

- enum [Colortype](#) { [primary](#) , [secondary](#) , [primary_accent](#) , [secondary_accent](#) }
An enumeration representing different color types for rendering.

Functions

- [Color_theme Cinit](#) ()
Initializes a [Color_theme](#) with a dynamically generated color scheme.

5.1.1 Detailed Description

This file contains color-related structures and functions.

5.1.2 Enumeration Type Documentation

5.1.2.1 Colortype

enum [Colortype](#)

An enumeration representing different color types for rendering.

Enumerator

primary	Primary color type
secondary	Secondary color type
primary_accent	Primary accent color type
secondary_accent	Secondary accent color type

5.1.3 Function Documentation

5.1.3.1 Cinit()

```
Color_theme Cinit ( )
```

Initializes a [Color_theme](#) with a dynamically generated color scheme.

Remarks

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.](#)

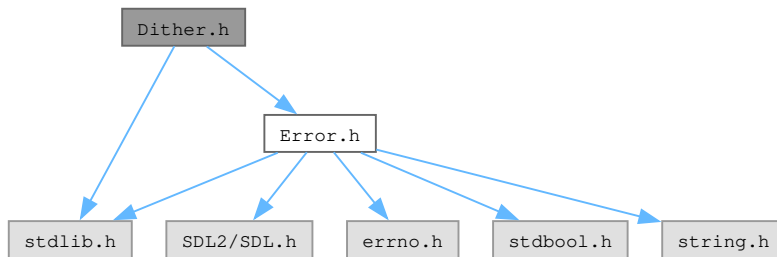
```
00001 #ifndef COLOR_H
00002 #define COLOR_H
00003
00004 #include <SDL2/SDL.h>
00005 #include <stdlib.h>
00006 #include <time.h>
00007 #include <math.h>
00008
00018 typedef enum Colortype
00019 {
00020     primary,
00021     secondary,
00022     primary_accent,
00023     secondary_accent
00024 } Colortype;
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
00047 Color_theme Cinit ();
00048
00049 #endif
```

5.3 Dither.h File Reference

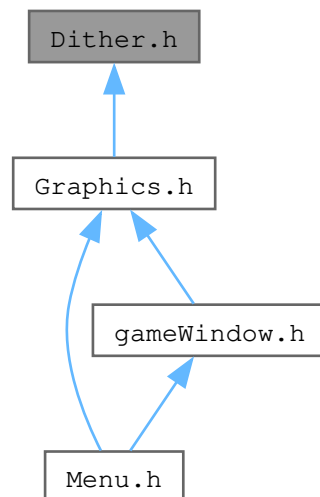
This file contains functions for generating and deallocating a Bayer matrix for ordered dithering.

```
#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

This file contains functions for generating and deallocating a Bayer matrix for ordered dithering.

Pre-renders cells in the graphics window.

Parameters

<i>window</i>	Pointer to the game window. Must not be NULL.
---------------	-----------------------------------------------

Returns

The texture of the pre-rendered cells.

Remarks

It uses the ordered dithering algorithm to render the fading effect.

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.

Remarks

This function deallocates the memory used by a Bayer matrix that was generated by Dgenerate_bayer_matrix.

Parameters

<i>matrix</i>	The Bayer matrix to be freed. It should be a valid pointer to a Bayer matrix generated by Dgenerate_bayer_matrix.
---------------	-------------------------------------------------------------------------------------------------------------------

5.3.2.2 Dgenerate_bayer_matrix()

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

Generates a Bayer matrix for ordered dithering.

Remarks

This function generates a Bayer matrix of size $n \times n$. The generated matrix is used for ordered dithering.

Parameters

n	The side length of the matrix. It should be a power of 2.
-----	-----------------------------------------------------------

Returns

A new Bayer matrix of size $n \times n$. Memory deallocation with `Dfree_bayer_matrix` is the caller's responsibility.

5.4 Dither.h

[Go to the documentation of this file.](#)

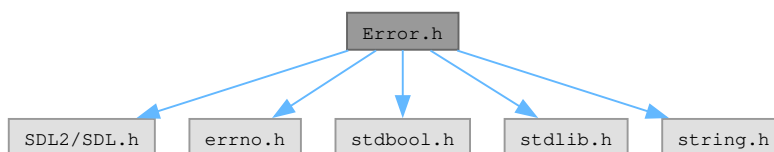
```
00001
00006 #ifndef DITHER_H
00007 #define DITHER_H
00008
00009 #include <stdlib.h>
00010 #include "Error.h"
00011
00020 size_t **Dgenerate_bayer_matrix(size_t n);
00021
00029 void Dfree_bayer_matrix(size_t **matrix);
00030
00031 #endif
```

5.5 Error.h File Reference

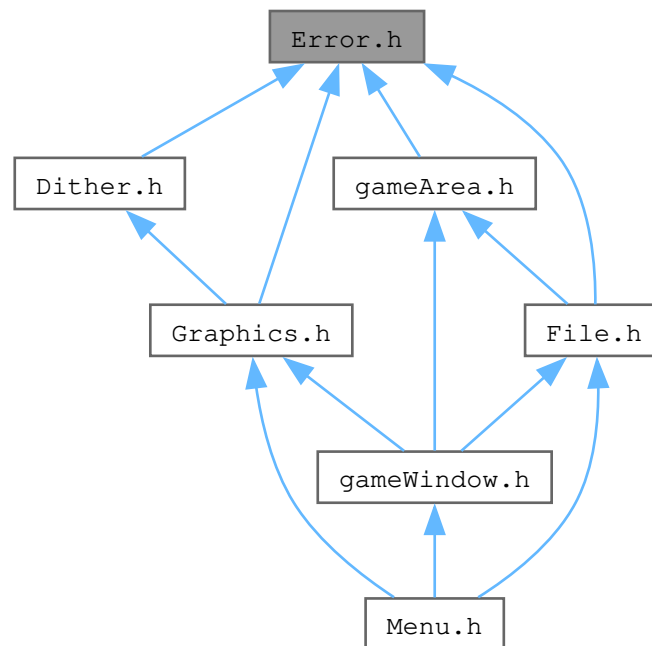
This file contains error handling macros and functions.

```
#include <SDL2/SDL.h>
#include <errno.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for Error.h:



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



Macros

- `#define ErrorIfTrue(test, error_msg) ErrorIfTrue_with_params(test, error_msg, __FILE__, __LINE__);`
Checks if the test is true and if so, triggers an error with the provided message.
- `#define ErrorIfNull(ptr, error_msg) ErrorIfTrue_with_params(ptr == NULL, error_msg, __FILE__, __LINE__);`
Checks if the pointer is null and if so, triggers an error with the provided message.
- `#define ErrorIfSdl(func_with_negative_error) ErrorIfTrue_with_params(func_with_negative_error < 0, "SDL hiba!", __FILE__, __LINE__);`
Checks if the SDL function returned a negative error code and if so, triggers an SDL error.
- `#define ErrorIfNoMemory(ptr) ErrorIfTrue_with_params(ptr == NULL, "Nincs eleg memoria!", __FILE__, __LINE__);`
Checks if the pointer is null due to insufficient memory and if so, triggers an error.

Functions

- `void ErrorIfTrue_with_params (bool test, char *error_msg, char *FILE, int LINE)`
Checks if the test is true and if so, triggers an error with the provided message, file name, and line number.
- `void ErrorWarning (char *error_msg)`
Displays a warning message box and logs an error message.

5.5.1 Detailed Description

This file contains error handling macros and functions.

5.5.2 Macro Definition Documentation

5.5.2.1 ErrorIfNoMemory

```
#define ErrorIfNoMemory(  
    ptr ) ErrorIftrue_with_params(ptr == NULL, "Nincs eleg memoria!", __FILE__, __  
LINE__);
```

Checks if the pointer is null due to insufficient memory and if so, triggers an error.

Parameters

<i>ptr</i>	The pointer to check.
------------	-----------------------

5.5.2.2 ErrorIfNull

```
#define ErrorIfNull(  
    ptr,  
    error_msg ) ErrorIftrue_with_params(ptr == NULL, error_msg, __FILE__, __LINE__);
```

Checks if the pointer is null and if so, triggers an error with the provided message.

Parameters

<i>ptr</i>	The pointer to check.
<i>error_msg</i>	The error message to display if the pointer is null.

5.5.2.3 ErrorIfSdl

```
#define ErrorIfSdl(  
    func_with_negative_error ) ErrorIftrue_with_params(func_with_negative_error < 0,  
"SDL hiba!", __FILE__, __LINE__);
```

Checks if the SDL function returned a negative error code and if so, triggers an SDL error.

Parameters

<i>func_with_negative_error</i>	The SDL function to check.
---------------------------------	----------------------------

5.5.2.4 ErrorIfTrue

```
#define ErrorIfTrue(  
    test,  
    error_msg ) ErrorIftrue_with_params(test, error_msg, __FILE__, __LINE__);
```

Checks if the test is true and if so, triggers an error with the provided message.

Parameters

<i>test</i>	The condition to check.
<i>error_msg</i>	The error message to display if the test is true.

5.5.3 Function Documentation

5.5.3.1 ErrorIftrue_with_params()

```
void ErrorIftrue_with_params (
    bool test,
    char * error_msg,
    char * FILE,
    int LINE )
```

Checks if the test is true and if so, triggers an error with the provided message, file name, and line number.

Parameters

<i>test</i>	The condition to check.
<i>error_msg</i>	The error message to display if the test is true.
<i>FILE</i>	The file name where the error occurred.
<i>LINE</i>	The line number where the error occurred.

5.5.3.2 ErrorWarning()

```
void ErrorWarning (
    char * error_msg )
```

Displays a warning message box and logs an error message.

Parameters

<i>error_msg</i>	The error message to be displayed.
------------------	------------------------------------

5.6 Error.h

[Go to the documentation of this file.](#)

```
00001
00006 #ifndef ERROR_H
00007 #define ERROR_H
00008
00009 #include <SDL2/SDL.h>
00010 #include <errno.h>
00011 #include <stdbool.h>
00012 #include <stdlib.h>
00013 #include <string.h>
00014
00021 #define ErrorIftrue(test, error_msg) ErrorIftrue_with_params(test, error_msg, __FILE__, __LINE__);
00022
```

```

00029 #define ErrorIFnull(ptr, error_msg) ErrorIFtrue_with_params(ptr == NULL, error_msg, __FILE__,
    __LINE__);
00030
00036 #define ErrorIFSdl(func_with_negative_error) ErrorIFtrue_with_params(func_with_negative_error < 0,
    "SDL hiba!", __FILE__, __LINE__);
00037
00043 #define ErrorIFnoMemory(ptr) ErrorIFtrue_with_params(ptr == NULL, "Nincs eleg memoria!", __FILE__,
    __LINE__);
00044
00052 void ErrorIFtrue_with_params(bool test, char* error_msg, char* FILE, int LINE);
00053
00058 void ErrorWarning(char* error_msg);
00059 #endif

```

5.7 File.h File Reference

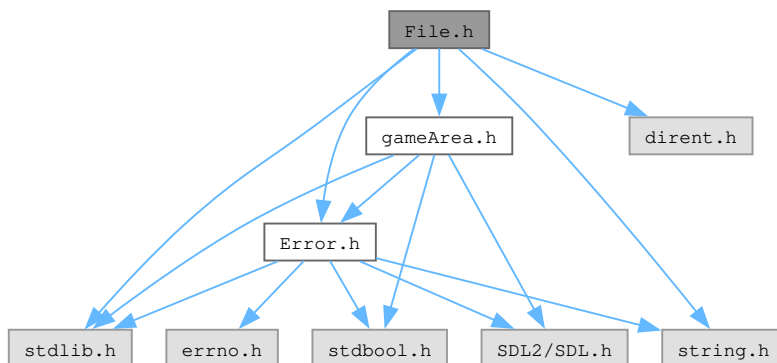
File operations for the game.

```

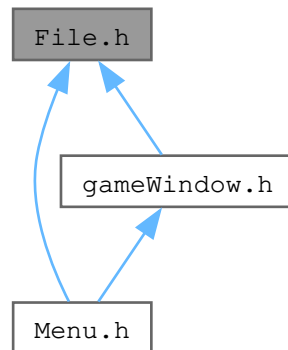
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include "Error.h"
#include "gameArea.h"

```

Include dependency graph for File.h:



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



Data Structures

- struct [Fgame_file](#)
Structure representing a game file.

Macros

- `#define SAVES_FOLDER "saved/"`
Directory for saved games.

Typedefs

- typedef struct [Fgame_file](#) **Fgame_file**
Structure representing a game file.

Functions

- [gameArea Fopen](#) (char *path)
Opens a game file.
- void [Fsave](#) (char *path, [gameArea](#) *gamearea)
Saves a game area to a file.
- size_t [Flist](#) ([Fgame_file](#) games[], size_t max_count)
Lists game files.

5.7.1 Detailed Description

File operations for the game.

5.7.2 Function Documentation

5.7.2.1 Flist()

```
size_t Flist (
    Fgame_file games[],
    size_t max_count )
```

Lists game files.

Parameters

<i>games</i>	Array of game files. Must not be NULL.
<i>max_count</i>	Maximum number of game files to list.

Returns

The number of game files listed.

5.7.2.2 Fopen()

```
gameArea Fopen (
    char * path )
```

Opens a game file.

Parameters

<i>path</i>	Path to the game file. Must be a valid path.
-------------	----------------------------------------------

Returns

The game area.

5.7.2.3 Fsave()

```
void Fsave (
    char * path,
    gameArea * gamearea )
```

Saves a game area to a file.

Parameters

<i>path</i>	Path to the game file. Should be a valid path.
<i>gamearea</i>	Pointer to the game area to save.

5.8 File.h

[Go to the documentation of this file.](#)

```

00001
00006 #ifndef FILE_H
00007 #define FILE_H
00008
00009 #include <stdlib.h>
00010 #include <dirent.h>
00011 #include <string.h>
00012
00013 #include "Error.h"
00014 #include "gameArea.h"
00015
00016 #define SAVES_FOLDER "saved/"
00017
00021 typedef struct Fgame_file{
00022     char *path;
00023     SDL_Rect location;
00024 } Fgame_file;
00025
00031 gameArea Fopen(char *path);
00032
00038 void Fsave(char *path, gameArea *gamearea);
00039
00046 size_t Flist(Fgame_file games[], size_t max_count);
00047
00048 #endif

```

5.9 gameArea.h File Reference

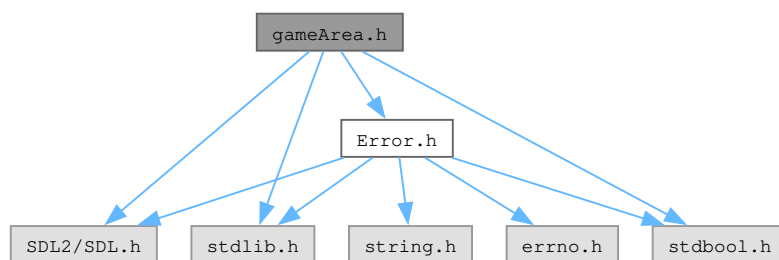
This file contains the structures and functions for the [gameArea](#), where the simulation takes place and the cells live.

```

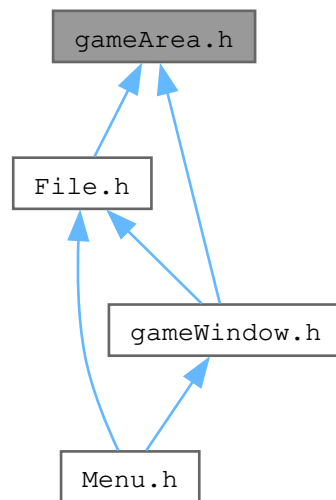
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdbool.h>
#include "Error.h"

```

Include dependency graph for gameArea.h:



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



Data Structures

- struct [gameArea](#)
Represents the game area and its properties.

Typedefs

- typedef struct [gameArea](#) **gameArea**

Functions

- [gameArea Anew](#) (size_t width, size_t height)
Creates a new game area.
- void [Aclear](#) ([gameArea](#) *gamearea)
Clears the game area.
- void [Afree](#) ([gameArea](#) *gamearea)
Frees the memory allocated for the game area.
- ssize_t [Agetage](#) (uint8_t cell)
Gets the age of a cell.
- void [Astep](#) ([gameArea](#) *A)
Advances the simulation by one step.
- bool [Aback](#) ([gameArea](#) *A)
Steps back the simulation by one step.
- void [Aflipcell](#) ([gameArea](#) *A, double x, double y)
Flips a cell in the game area.

5.9.1 Detailed Description

This file contains the structures and functions for the `gameArea`, where the simulation takes place and the cells live.

5.9.2 Function Documentation

5.9.2.1 Aback()

```
bool Aback (
    gameArea * A )
```

Steps back the simulation by one step.

Parameters

<i>A</i>	Pointer to the game area to step back. Must not be NULL.
----------	----------------------------------------------------------

Returns

True if successful, false otherwise.

5.9.2.2 Aclear()

```
void Aclear (
    gameArea * gamearea )
```

Clears the game area.

Parameters

<i>gamearea</i>	Pointer to the game area to clear. Must not be NULL.
-----------------	------------------------------------------------------

5.9.2.3 Aflipcell()

```
void Aflipcell (
    gameArea * A,
    double x,
    double y )
```

Flips a cell in the game area.

Parameters

<i>A</i>	Pointer to the game area. Must not be NULL.
<i>x</i>	The x-coordinate of the cell to flip.
<i>y</i>	The y-coordinate of the cell to flip.

Remarks

If the coordinates are out of bounds, the function does nothing.

5.9.2.4 Afree()

```
void Afree (
    gameArea * gamearea )
```

Frees the memory allocated for the game area.

Parameters

<i>gamearea</i>	Pointer to the game area to free. Must not be NULL.
-----------------	-----------------------------------------------------

5.9.2.5 Agetage()

```
ssize_t Agetage (
    uint8_t cell )
```

Gets the age of a cell.

Parameters

<i>cell</i>	The cell to get the age of.
-------------	-----------------------------

Returns

The age of the cell.

5.9.2.6 Anew()

```
gameArea Anew (
    size_t width,
    size_t height )
```

Creates a new game area.

Parameters

<i>width</i>	Width of the game area.
<i>height</i>	Height of the game area.

Returns

A new game area.

5.9.2.7 Astep()

```
void Astep (
    gameArea * A )
```

Advances the simulation by one step.

Parameters

A	Pointer to the game area to step. Must not be NULL.
---	-----------------------------------------------------

5.10 gameArea.h

[Go to the documentation of this file.](#)

```
00001
00007 #ifndef GAMEAREA_H
00008 #define GAMEAREA_H
00009
00010 #include <SDL2/SDL.h>
00011 #include <stdlib.h>
00012 #include <stdbool.h>
00013
00014 #include "Error.h"
00015
00022 typedef struct gameArea {
00023     size_t w;
00024     size_t h;
00025     uint8_t **area;
00026     uint8_t history_lenght;
00027 } gameArea;
00028
00035 gameArea Anew(size_t width, size_t height);
00036
00041 void Aclear(gameArea *gamearea);
00042
00047 void Afree(gameArea *gamearea);
00048
00054 ssize_t Agetage(uint8_t cell);
00055
00060 void Astep(gameArea *A);
00061
00067 bool Aback(gameArea *A);
00068
00076 void Aflipcell(gameArea *A, double x, double y);
00077
00078 #endif
```

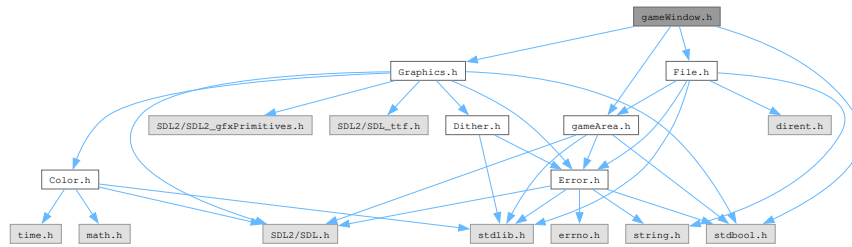
5.11 gameWindow.h File Reference

This file contains the structures and functions for the game window.

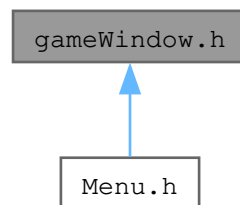
```
#include <stdbool.h>
#include "Graphics.h"
#include "gameArea.h"
```

```
#include "File.h"
```

Include dependency graph for gameWindow.h:



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



Data Structures

- struct [gameWindow](#)
Represents the game window and its properties.

Typedefs

- typedef struct [gameWindow](#) **gameWindow**

Functions

- [gameWindow Winit](#) ([gameArea](#) A, char *name)
Initializes a new game window.
- void [Wclose](#) ([gameWindow](#) *game)
Closes the game window.
- void [Wclick](#) ([gameWindow](#) *game, int x, int y)
Handles a click event in the game window.
- void [Wdraw](#) ([gameWindow](#) *game, bool all_cells)
Draws the game window.
- void [Wzoom](#) ([gameWindow](#) *game, double wheel, int x, int y)
Zooms the game window.
- void [Wresetzoom](#) ([gameWindow](#) *game)
Resets the zoom level of the game window.
- void [Wevent](#) ([gameWindow](#) *game, SDL_Event *e)
Handles an event in the game window.

5.11.1 Detailed Description

This file contains the structures and functions for the game window.

5.11.2 Function Documentation

5.11.2.1 Wclick()

```
void Wclick (
    gameWindow * game,
    int x,
    int y )
```

Handles a click event in the game window.

Parameters

<i>game</i>	Pointer to the game window. Must not be NULL.
<i>x</i>	The x-coordinate of the click.
<i>y</i>	The y-coordinate of the click.

5.11.2.2 Wclose()

```
void Wclose (
    gameWindow * game )
```

Closes the game window.

Parameters

<i>game</i>	Pointer to the game window to close. Must not be NULL.
-------------	--------------------------------------------------------

5.11.2.3 Wdraw()

```
void Wdraw (
    gameWindow * game,
    bool all_cells )
```

Draws the game window.

Parameters

<i>game</i>	Pointer to the game window to draw. Must not be NULL.
<i>all_cells</i>	Whether to draw all cells or just the ones that changed.

5.11.2.4 Wevent()

```
void Wevent (
    gameWindow * game,
    SDL_Event * e )
```

Handles an event in the game window.

Parameters

<i>game</i>	Pointer to the game window to handle event. Must not be NULL.
<i>e</i>	The event to handle.

5.11.2.5 Winit()

```
gameWindow Winit (
    gameArea A,
    char * name )
```

Initializes a new game window.

Parameters

<i>A</i>	The game area. Takes ownership of the game area and frees it when the game window is closed. Must not be NULL.
<i>name</i>	The name of the game window. Must not be NULL.

Returns

A new game window.

5.11.2.6 Wresetzoom()

```
void Wresetzoom (
    gameWindow * game )
```

Resets the zoom level of the game window.

Parameters

<i>game</i>	Pointer to the game window to reset zoom. Must not be NULL.
-------------	-------------------------------------------------------------

5.11.2.7 Wzoom()

```
void Wzoom (
    gameWindow * game,
    double wheel,
```

```

    int x,
    int y )

```

Zooms the game window.

Parameters

<i>game</i>	Pointer to the game window to zoom. Must not be NULL.
<i>wheel</i>	The amount to zoom.
<i>x</i>	The x-coordinate of the zoom center.
<i>y</i>	The y-coordinate of the zoom center.

5.12 gameWindow.h

[Go to the documentation of this file.](#)

```

00001
00006 #ifndef GAMEWINDOW_H
00007 #define GAMEWINDOW_H
00008
00009 #include <stdbool.h>
00010 #include "Graphics.h"
00011 #include "gameArea.h"
00012 #include "File.h"
00013
00019 typedef struct gameWindow {
00020     gameArea A;
00021     Gwindow G;
00022     char *name;
00023     SDL_Texture *pre_rendered_cells;
00024     double zoom;
00025     ssize_t x_screen_offset;
00026     ssize_t y_screen_offset;
00027     SDL_TimerID autoplay_id;
00028     Uint32 autoplay_delay;
00029 } gameWindow;
00030
00038 gameWindow Winit(gameArea A, char *name);
00039
00044 void Wclose(gameWindow *game);
00045
00052 void Wclick(gameWindow *game, int x, int y);
00053
00059 void Wdraw(gameWindow *game, bool all_cells);
00060
00068 void Wzoom(gameWindow *game, double wheel, int x, int y);
00069
00074 void Wresetzoom(gameWindow *game);
00075
00081 void Wevent(gameWindow *game, SDL_Event *e);
00082
00083 #endif

```

5.13 Graphics.h File Reference

This file contains code used for graphics.

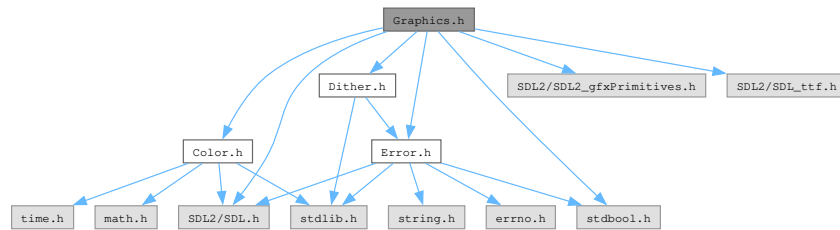
```

#include <SDL2/SDL.h>
#include <SDL2/SDL2_gfxPrimitives.h>
#include <SDL2/SDL_ttf.h>
#include <stdbool.h>
#include "Color.h"
#include "Dither.h"

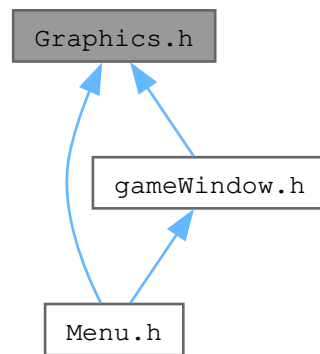
```

```
#include "Error.h"
```

Include dependency graph for Graphics.h:



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



Data Structures

- struct [Gwindow](#)
Represents the graphics window and its properties.

Macros

- #define [CELL_SIZE](#) 8
The size of a rendered cell in pixels.

Typedefs

- typedef struct [Gwindow](#) **Gwindow**

Functions

- void [Ginit](#) ()
Initializes SDL2.
- [Gwindow Gnew](#) (char title[], int width, int height, bool resizable)
Creates a new window.
- void [Gclose](#) ([Gwindow](#) *window)
Closes the graphics window.
- void [Gquit](#) ()
Quits the SDL2 application.
- void [Gset_color](#) ([Gwindow](#) *window, SDL_Color col)
Sets the color of the renderer.
- void [Gfill_background](#) ([Gwindow](#) *window)
Fills the background of the [Menu](#).
- void [Gprint_title](#) ([Gwindow](#) *window)
Prints the title of the game.
- SDL_Rect [Gprint](#) ([Gwindow](#) *window, char *text, SDL_Rect *location, [Colortype](#) col)
Prints text in the graphics window.
- void [Gtextbox](#) ([Gwindow](#) *window, char *text, SDL_Rect *location, [Colortype](#) col, size_t border_width)
Creates a textbox in the graphics window.
- SDL_Texture * [Gpre_render_cells](#) ([Gwindow](#) *window)
- void [Ginput_text](#) ([Gwindow](#) *window, char *dest, size_t lenght, SDL_Rect bounding_box, bool is_file_name)
Handles text input in the graphics window.

5.13.1 Detailed Description

This file contains code used for graphics.

5.13.2 Macro Definition Documentation

5.13.2.1 CELL_SIZE

```
#define CELL_SIZE 8
```

The size of a rendered cell in pixels.

It should be a power of 2. Smaller values will result in better performance, but worse quality.

5.13.3 Function Documentation

5.13.3.1 Gclose()

```
void Gclose (
    Gwindow * window )
```

Closes the graphics window.

Parameters

<i>window</i>	Pointer to the graphics window to close. Must not be NULL.
---------------	------------------------------------------------------------

5.13.3.2 Gfill_background()

```
void Gfill_background (
    Gwindow * window )
```

Fills the background of the [Menu](#).

Parameters

<i>window</i>	Pointer to the window to fill background. Must not be NULL.
---------------	-------------------------------------------------------------

5.13.3.3 Ginit()

```
void Ginit ( )
```

Initializes SDL2.

Warning

This function must be called before any other function.

5.13.3.4 Ginput_text()

```
void Ginput_text (
    Gwindow * window,
    char * dest,
    size_t lenght,
    SDL_Rect bounding_box,
    bool is_file_name )
```

Handles text input in the graphics window.

Parameters

<i>window</i>	Pointer to the window to handle text input. Must not be NULL.
<i>dest</i>	The destination for the input text.
<i>lenght</i>	The length of the input text.
<i>bounding_box</i>	The bounding box for the input text.
<i>is_file_name</i>	Whether the input text is a file name or not. If it is, it will append ".con" to the end, and the destination must be 4 bytes longer than lenght.

5.13.3.5 Gnew()

```
Gwindow Gnew (
    char title[],
    int width,
    int height,
    bool resizable )
```

Creates a new window.

Parameters

<i>title</i>	The title of the window. Must not be NULL.
<i>width</i>	The width of the window.
<i>height</i>	The height of the window.
<i>resizable</i>	Whether the window is resizable or not.

Returns

A new window.

5.13.3.6 Gprint()

```
SDL_Rect Gprint (
    Gwindow * window,
    char * text,
    SDL_Rect * location,
    Colortype col )
```

Prints text in the graphics window.

Parameters

<i>window</i>	Pointer to the window to print text. Must not be NULL.
<i>text</i>	The text to print. Empty string for no text. Must not be NULL.
<i>location</i>	The location to print the text.
<i>col</i>	The color of the text.

Returns

The rectangle where the text was printed.

5.13.3.7 Gprint_title()

```
void Gprint_title (
    Gwindow * window )
```

Prints the title of the game.

Parameters

<i>window</i>	Pointer to the window to print title. Must not be NULL.
---------------	---------------------------------------------------------

5.13.3.8 Gquit()

```
void Gquit ( )
```

Quits the SDL2 application.

Remarks

exit() should be called after this function.

5.13.3.9 Gset_color()

```
void Gset_color (
    Gwindow * window,
    SDL_Color col )
```

Sets the color of the renderer.

Parameters

<i>window</i>	Pointer to the window to set color. Must not be NULL.
<i>col</i>	The color to set.

5.13.3.10 Gtextbox()

```
void Gtextbox (
    Gwindow * window,
    char * text,
    SDL_Rect * location,
    Colortype col,
    size_t border_width )
```

Creates a textbox in the graphics window.

Parameters

<i>window</i>	Pointer to the window to create textbox. Must not be NULL.
<i>text</i>	The text for the textbox. Empty string for no text. Must not be NULL.
<i>location</i>	The location for the textbox.
<i>col</i>	The color of the textbox, only accepts primary or secondary.
<i>border_width</i>	The width of the border of the textbox.

5.14 Graphics.h

[Go to the documentation of this file.](#)

```

00001
00006 #ifndef GRAPHICS_H
00007 #define GRAPHICS_H
00008
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL2_gfxPrimitives.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <stdbool.h>
00013
00014 #include "Color.h"
00015 #include "Dither.h"
00016 #include "Error.h"
00017
00023 #define CELL_SIZE 8
00024
00029 typedef struct Gwindow {
00030     SDL_Window *win;
00031     SDL_Renderer *ren;
00032     size_t w, h;
00033     TTF_Font *font_big;
00034     TTF_Font *font_reg;
00035     Color_theme colors;
00036 } Gwindow;
00037
00042 void Ginit();
00043
00052 Gwindow Gnew(char title[], int width, int height, bool resizable);
00053
00058 void Gclose(Gwindow *window);
00059
00064 void Gquit();
00065
00071 void Gset_color(Gwindow *window, SDL_Color col);
00072
00077 void Gfill_background(Gwindow *window);
00078
00083 void Gprint_title(Gwindow *window);
00084
00093 SDL_Rect Gprint(Gwindow *window, char *text, SDL_Rect *location, Color_type col);
00094
00103 void Gtextbox(Gwindow *window, char *text, SDL_Rect *location, Color_type col, size_t border_width);
00104
00112 SDL_Texture *Gpre_render_cells(Gwindow *window);
00113
00123 void Ginput_text(Gwindow *window, char *dest, size_t length, SDL_Rect bounding_box, bool
is_file_name);
00124
00125 #endif

```

5.15 Menu.h File Reference

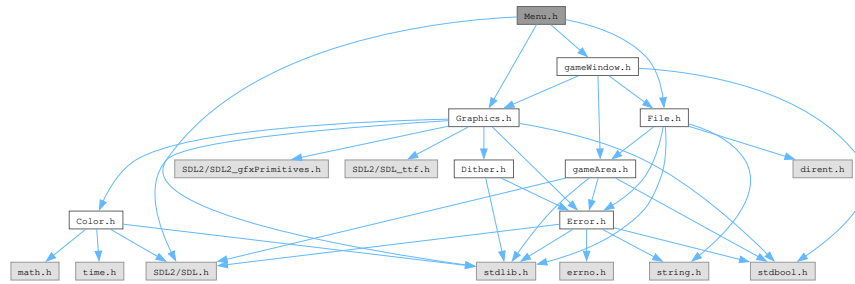
This file contains the structures and functions for the menu window.

```

#include <stdlib.h>
#include "Graphics.h"
#include "File.h"
#include "gameWindow.h"

```

Include dependency graph for Menu.h:



Data Structures

- struct [uj_jatek_input](#)
Represents the input for a new game.
- struct [Menu](#)
Represents the menu and its properties.

Macros

- `#define MAX_SAVES 13`
- `#define INPUT_MAX_LENIGHT 15`

Typedefs

- typedef struct [uj_jatek_input](#) **uj_jatek_input**
- typedef struct [Menu](#) **Menu**

Functions

- [Menu Minit](#) ()
Initializes the menu.
- void [Mclose](#) ([Menu](#) *menu)
Closes the menu.
- void [Mclick](#) ([Menu](#) *menu, int x, int y)
Handles a click event in the menu.
- void [Mevent](#) ([Menu](#) *menu, SDL_Event *e)
Handles an event in the menu. Must not be NULL.

5.15.1 Detailed Description

This file contains the structures and functions for the menu window.

5.15.2 Function Documentation

5.15.2.1 Mclick()

```
void Mclick (
    Menu * menu,
    int x,
    int y )
```

Handles a click event in the menu.

Parameters

<i>menu</i>	Pointer to the menu. Must not be NULL.
<i>x</i>	The x-coordinate of the click.
<i>y</i>	The y-coordinate of the click.

5.15.2.2 Mclose()

```
void Mclose (
    Menu * menu )
```

Closes the menu.

Parameters

<i>menu</i>	Pointer to the menu to close. Must not be NULL.
-------------	-------------------------------------------------

5.15.2.3 Mevent()

```
void Mevent (
    Menu * menu,
    SDL_Event * e )
```

Handles an event in the menu. Must not be NULL.

Parameters

<i>menu</i>	Pointer to the menu to handle event.
<i>e</i>	The event to handle. Must not be NULL.

5.15.2.4 Minit()

```
Menu Minit ( )
```

Initializes the menu.

Returns

The menu.

5.16 Menu.h

[Go to the documentation of this file.](#)

```
00001
00006 #ifndef MENU_H
00007 #define MENU_H
00008
00009 #include <stdlib.h>
```



```
00010
00011 #include "Graphics.h"
00012 #include "File.h"
00013 #include "gameWindow.h"
00014
00015 #define MAX_SAVES 13
00016 #define INPUT_MAX LENGHT 15
00017
00022 typedef struct uj_jatek_input{
00023     char name[INPUT_MAX LENGHT+4];
00024     SDL_Rect name_rct;
00025     char width[INPUT_MAX LENGHT];
00026     SDL_Rect width_rct;
00027     char height[INPUT_MAX LENGHT];
00028     SDL_Rect height_rct;
00029     SDL_Rect button;
00030 } uj_jatek_input;
00031
00037 typedef struct Menu{
00038     Gwindow G;
00039     Fgame_file *saves;
00040     size_t save_cnt;
00041     gameWindow game_open;
00042     uj_jatek_input text_input;
00043 } Menu;
00044
00049 Menu Minit();
00050
00055 void Mclose(Menu *menu);
00056
00063 void Mclick(Menu *menu, int x, int y);
00064
00070 void Mevent(Menu *menu, SDL_Event *e);
00071
00072 #endif
```


Index

Aback
 gameArea.h, [27](#)

Aclear
 gameArea.h, [27](#)

Aflipcell
 gameArea.h, [27](#)

Afree
 gameArea.h, [28](#)

Agetage
 gameArea.h, [28](#)

Anew
 gameArea.h, [28](#)

Astep
 gameArea.h, [28](#)

bg
 Color_theme, [7](#)

CELL_SIZE
 Graphics.h, [35](#)

Cinit
 Color.h, [15](#)

Color.h, [13](#), [15](#)
 Cinit, [15](#)
 Colortype, [14](#)
 primary, [15](#)
 primary_accent, [15](#)
 secondary, [15](#)
 secondary_accent, [15](#)

Color_theme, [7](#)
 bg, [7](#)
 prim, [7](#)
 primacc, [7](#)
 sec, [8](#)
 secacc, [8](#)

Colortype
 Color.h, [14](#)

Dfree_bayer_matrix
 Dither.h, [17](#)

Dgenerate_bayer_matrix
 Dither.h, [17](#)

Dither.h, [16](#), [18](#)
 Dfree_bayer_matrix, [17](#)
 Dgenerate_bayer_matrix, [17](#)

Error.h, [18](#), [21](#)
 ErrorIFnoMemory, [20](#)
 ErrorIFnull, [20](#)
 ErrorIFsdl, [20](#)
 ErrorIFtrue, [20](#)
 ErrorIFtrue_with_params, [21](#)
 ErrorWarning, [21](#)

ErrorIFnoMemory
 Error.h, [20](#)

ErrorIFnull
 Error.h, [20](#)

ErrorIFsdl
 Error.h, [20](#)

ErrorIFtrue
 Error.h, [20](#)

ErrorIFtrue_with_params
 Error.h, [21](#)

ErrorWarning
 Error.h, [21](#)

Fgame_file, [8](#)

File.h, [22](#), [25](#)
 Flist, [24](#)
 Fopen, [24](#)
 Fsave, [24](#)

Flist
 File.h, [24](#)

Fopen
 File.h, [24](#)

Fsave
 File.h, [24](#)

Game of Life by David Zoller, [1](#)

gameArea, [8](#)

gameArea.h, [25](#), [29](#)
 Aback, [27](#)
 Aclear, [27](#)
 Aflipcell, [27](#)
 Afree, [28](#)
 Agetage, [28](#)
 Anew, [28](#)
 Astep, [28](#)

gameWindow, [9](#)

gameWindow.h, [29](#), [33](#)
 Wclick, [31](#)
 Wclose, [31](#)
 Wdraw, [31](#)
 Wevent, [31](#)
 Winit, [32](#)
 Wresetzoom, [32](#)
 Wzoom, [32](#)

Gclose
 Graphics.h, [35](#)

Gfill_background

- Graphics.h, 36
- Ginit
 - Graphics.h, 36
- Ginput_text
 - Graphics.h, 36
- Gnew
 - Graphics.h, 36
- Gprint
 - Graphics.h, 37
- Gprint_title
 - Graphics.h, 37
- Gquit
 - Graphics.h, 38
- Graphics.h, 33, 39
 - CELL_SIZE, 35
 - Gclose, 35
 - Gfill_background, 36
 - Ginit, 36
 - Ginput_text, 36
 - Gnew, 36
 - Gprint, 37
 - Gprint_title, 37
 - Gquit, 38
 - Gset_color, 38
 - Gtextbox, 38
- Gset_color
 - Graphics.h, 38
- Gtextbox
 - Graphics.h, 38
- Gwindow, 10
- Mclick
 - Menu.h, 41
- Mclose
 - Menu.h, 42
- Menu, 11
- Menu.h, 39, 42
 - Mclick, 41
 - Mclose, 42
 - Mevent, 42
 - Minit, 42
- Mevent
 - Menu.h, 42
- Minit
 - Menu.h, 42
- prim
 - Color_theme, 7
- primacc
 - Color_theme, 7
- primary
 - Color.h, 15
- primary_accent
 - Color.h, 15
- sec
 - Color_theme, 8
- secacc
 - Color_theme, 8
- secondary
 - Color.h, 15
- secondary_accent
 - Color.h, 15
- uj_jatek_input, 12
- Wclick
 - gameWindow.h, 31
- Wclose
 - gameWindow.h, 31
- Wdraw
 - gameWindow.h, 31
- Wevent
 - gameWindow.h, 31
- Winit
 - gameWindow.h, 32
- Wresetzoom
 - gameWindow.h, 32
- Wzoom
 - gameWindow.h, 32