

## Game of Life

Generated by Doxygen 1.9.8



<b>1 Game of Life by David Zoller</b>	<b>1</b>
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
<b>2 Data Structure Index</b>	<b>3</b>
2.1 Data Structures	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Data Structure Documentation</b>	<b>7</b>
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
<b>5 File Documentation</b>	<b>13</b>
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 Dither.h File Reference	15

5.2.1 Detailed Description	16
5.2.2 Function Documentation	16
5.2.2.1 Dfree_bayer_matrix()	16
5.2.2.2 Dgenerate_bayer_matrix()	17
5.3 Error.h File Reference	17
5.3.1 Detailed Description	18
5.3.2 Macro Definition Documentation	19
5.3.2.1 ErrorIFnoMemory	19
5.3.2.2 ErrorIFnull	19
5.3.2.3 ErrorIFsdl	19
5.3.2.4 ErrorIFtrue	19
5.3.3 Function Documentation	20
5.3.3.1 ErrorIFtrue_with_params()	20
5.3.3.2 ErrorWarning()	20
5.4 File.h File Reference	20
5.4.1 Detailed Description	22
5.4.2 Function Documentation	22
5.4.2.1 Flist()	22
5.4.2.2 Fopen()	22
5.4.2.3 Fsave()	23
5.5 gameArea.h File Reference	23
5.5.1 Detailed Description	24
5.5.2 Function Documentation	25
5.5.2.1 Aback()	25
5.5.2.2 Aclear()	25
5.5.2.3 Aflipcell()	25
5.5.2.4 Afree()	26
5.5.2.5 Agetage()	26
5.5.2.6 Anew()	26
5.5.2.7 Astep()	26
5.6 gameWindow.h File Reference	28
5.6.1 Detailed Description	29
5.6.2 Function Documentation	29
5.6.2.1 Wclick()	29
5.6.2.2 Wclose()	29
5.6.2.3 Wdraw()	30
5.6.2.4 Wevent()	30
5.6.2.5 Winit()	30
5.6.2.6 Wresetzoom()	30
5.6.2.7 Wzoom()	32
5.7 Graphics.h File Reference	32
5.7.1 Detailed Description	34

5.7.2 Macro Definition Documentation . . . . .	34
5.7.2.1 CELL_SIZE . . . . .	34
5.7.3 Function Documentation . . . . .	34
5.7.3.1 Gclose() . . . . .	34
5.7.3.2 Gfill_background() . . . . .	34
5.7.3.3 Ginit() . . . . .	34
5.7.3.4 Ginput_text() . . . . .	35
5.7.3.5 Gnew() . . . . .	35
5.7.3.6 Gpre_render_cells() . . . . .	35
5.7.3.7 Gprint() . . . . .	36
5.7.3.8 Gprint_title() . . . . .	36
5.7.3.9 Gquit() . . . . .	37
5.7.3.10 Gset_color() . . . . .	37
5.7.3.11 Gtextbox() . . . . .	37
5.8 Menu.h File Reference . . . . .	37
5.8.1 Detailed Description . . . . .	38
5.8.2 Function Documentation . . . . .	38
5.8.2.1 Mclick() . . . . .	38
5.8.2.2 Mclose() . . . . .	39
5.8.2.3 Mevent() . . . . .	39
5.8.2.4 Minit() . . . . .	39
<b>Index</b>	<b>41</b>



# 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: [https://en.wikipedia.org/wiki/Conway%27s\\_Game\\_of\\_Life](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)

## 1.2 Structure

The project is structured into several modules:

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

<a href="#">Color_theme</a>	A structure representing a color theme with five colors . . . . .	7
<a href="#">Fgame_file</a>	Structure representing a game file . . . . .	8
<a href="#">gameArea</a>	Represents the game area and its properties . . . . .	8
<a href="#">gameWindow</a>	Represents the game window and its properties . . . . .	9
<a href="#">Gwindow</a>	Represents the graphics window and its properties . . . . .	10
<a href="#">Menu</a>	Represents the menu and its properties . . . . .	11
<a href="#">uj_jatek_input</a>	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:

<a href="#">Color.h</a>	This file contains color-related structures and functions . . . . .	13
<a href="#">Dither.h</a>	This file contains functions for generating and deallocating a Bayer matrix for ordered dithering	15
<a href="#">Error.h</a>	This file contains error handling macros and functions . . . . .	17
<a href="#">File.h</a>	File operations for the game . . . . .	20
<a href="#">gameArea.h</a>	This file contains the structures and functions for the <a href="#">gameArea</a> , where the simulation takes place and the cells live . . . . .	23
<a href="#">gameWindow.h</a>	This file contains the structures and functions for the game window . . . . .	28
<a href="#">Graphics.h</a>	This file contains code used for graphics . . . . .	32
<a href="#">Menu.h</a>	This file contains the structures and functions for the menu window . . . . .	37



## 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.

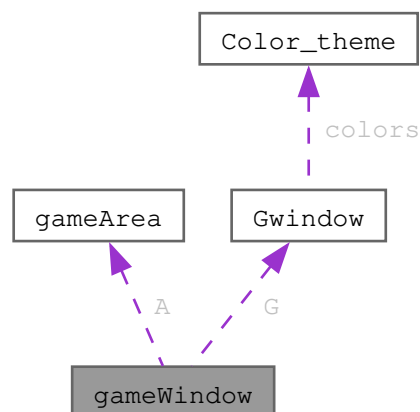
This structure should only be created with a function, and must be deleted with the Afree function.

## 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.

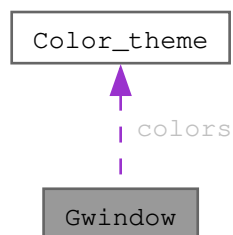
This structure should only be created with a function, and must be deleted with the Wclose function.

## 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`  
*The width of the window.*
- `size_t h`  
*The 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

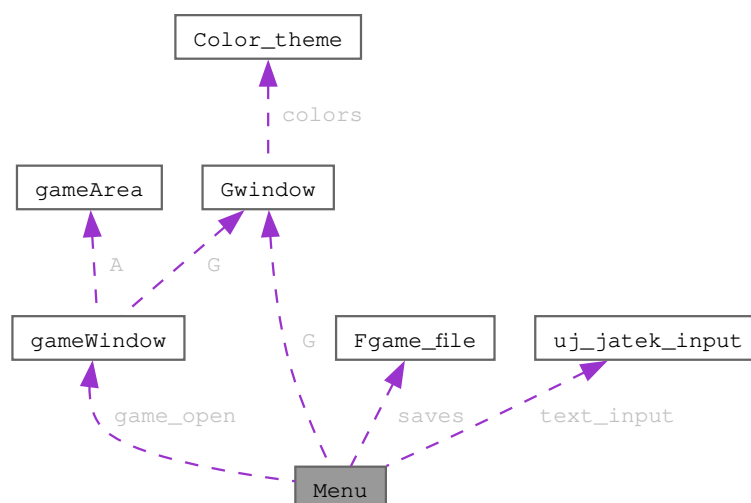
Represents the graphics window and its properties.

## 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

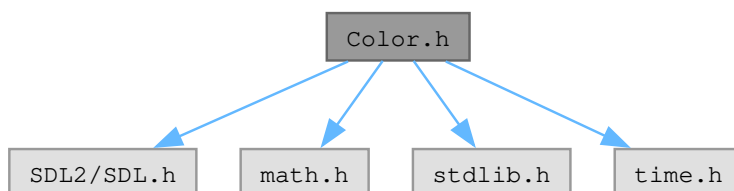
# File Documentation

### 5.1 Color.h File Reference

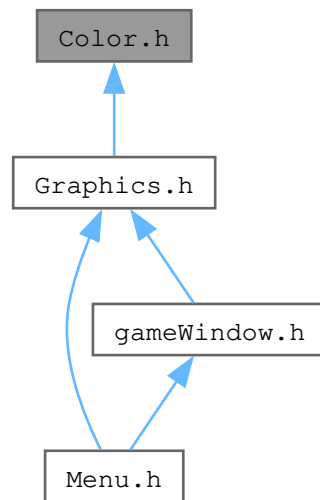
This file contains color-related structures and functions.

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

## 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.

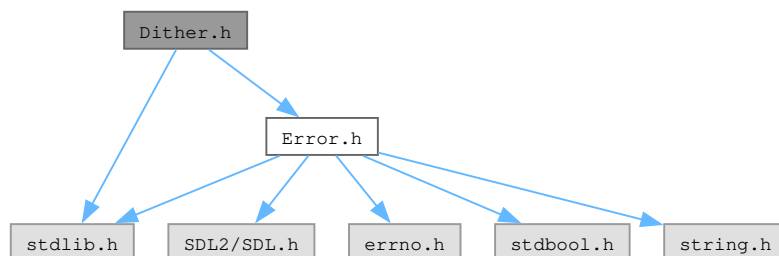
## 5.2 Dither.h File Reference

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

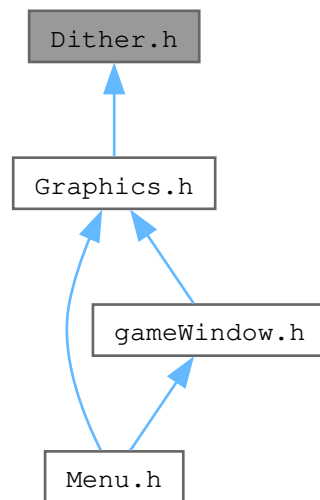
```
#include <stdlib.h>
```

```
#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.*

### 5.2.1 Detailed Description

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

### 5.2.2 Function Documentation

#### 5.2.2.1 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 <code>Dgenerate_bayer_matrix</code> .
---------------	---

### 5.2.2.2 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

<i>n</i>	The side length of the matrix. It should be a power of 2.
----------	---

## Returns

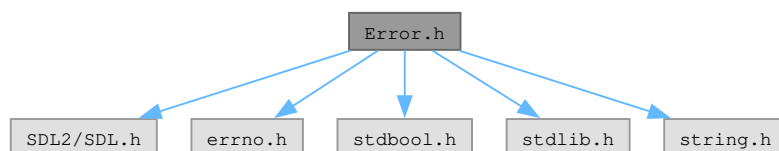
A new Bayer matrix of size  $n \times n$ . Memory deallocation with `Dfree_bayer_matrix` is the caller's responsibility.

## 5.3 Error.h File Reference

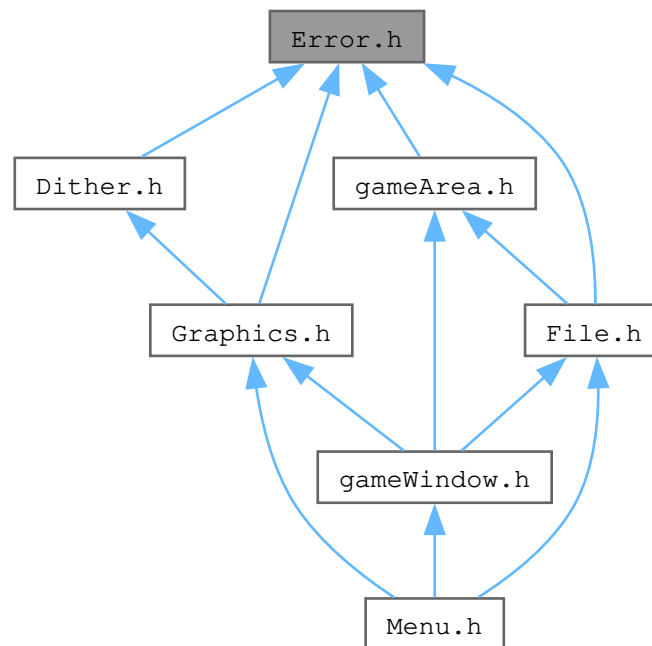
This file contains error handling macros and functions.

```
#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.

### 5.3.1 Detailed Description

This file contains error handling macros and functions.



## 5.3.2 Macro Definition Documentation

### 5.3.2.1 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.
------------	-----------------------

### 5.3.2.2 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.

### 5.3.2.3 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.
---------------------------------	----------------------------

### 5.3.2.4 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.

### 5.3.3 Function Documentation

#### 5.3.3.1 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.

#### 5.3.3.2 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.
------------------	------------------------------------

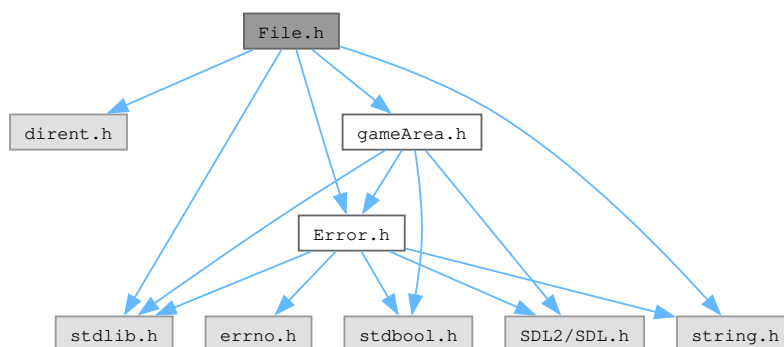
## 5.4 File.h File Reference

File operations for the game.

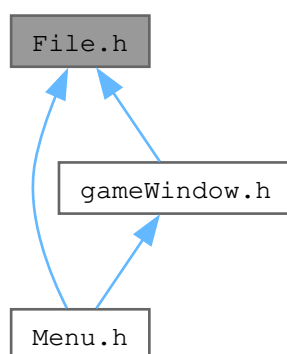
```
#include <dirent.h>
#include <stdlib.h>
#include <string.h>
#include "Error.h"
```

```
#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.*

## 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.*

### 5.4.1 Detailed Description

File operations for the game.

### 5.4.2 Function Documentation

#### 5.4.2.1 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.

#### 5.4.2.2 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.

### 5.4.2.3 Fsave()

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

Saves a game area to a file.

#### Parameters

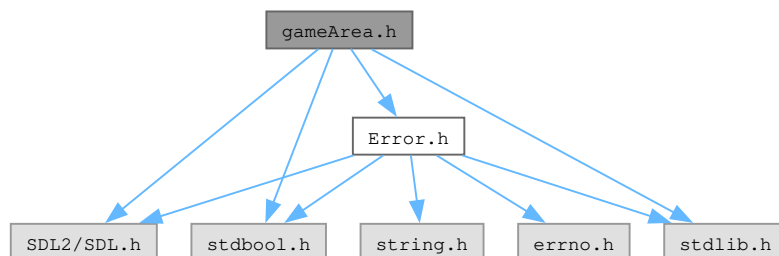
<i>path</i>	Path to the game file. Should be a valid path.
<i>gamearea</i>	Pointer to the game area to save.

## 5.5 gameArea.h File Reference

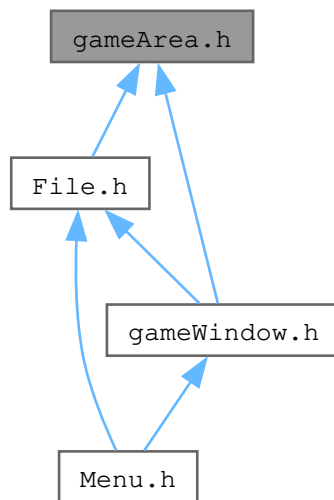
This file contains the structures and functions for the [gameArea](#), where the simulation takes place and the cells live.

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

## 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.*

### 5.5.1 Detailed Description

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

## 5.5.2 Function Documentation

### 5.5.2.1 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.

### 5.5.2.2 Aclear()

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

Clears the game area.

#### Parameters

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

### 5.5.2.3 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.

#### 5.5.2.4 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.
-----------------	---

#### 5.5.2.5 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.

#### 5.5.2.6 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.

#### 5.5.2.7 Astep()

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



Advances the simulation by one step.

## Parameters

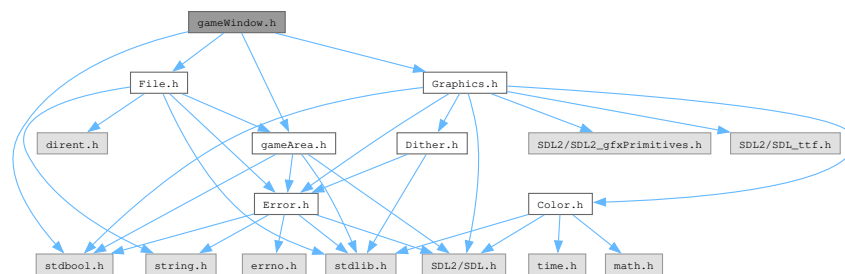
A	Pointer to the game area to step. Must not be NULL.
---	---

## 5.6 gameWindow.h File Reference

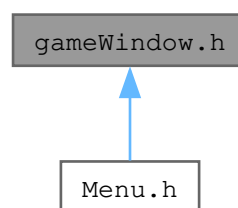
This file contains the structures and functions for the game window.

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

## 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.*

### 5.6.1 Detailed Description

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

### 5.6.2 Function Documentation

#### 5.6.2.1 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.

#### 5.6.2.2 Wclose()

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

Closes the game window.

##### Parameters

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

### 5.6.2.3 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.

### 5.6.2.4 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.

### 5.6.2.5 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.

### 5.6.2.6 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.
-------------	---

## 5.6.2.7 Wzoom()

```
void Wzoom (
    gameWindow * game,
    double wheel,
    int x,
    int y )
```

Zooms the game window.

## Parameters

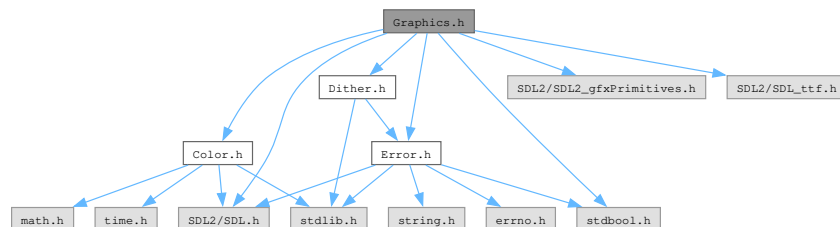
<i>game</i>	Pointer to the game window to zoom. Must not be NULL.
<i>wheel</i>	The amount to zoom.
<i>x</i>	The x-coordinate of the zoom center.
<i>y</i>	The y-coordinate of the zoom center.

## 5.7 Graphics.h File Reference

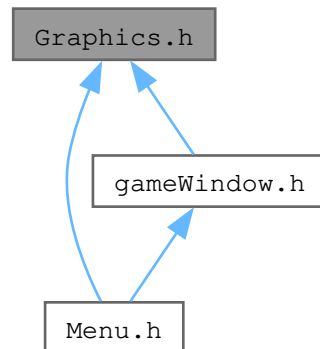
This file contains code used for graphics.

```
#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.*

### 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)  
*Pre-renders cells in the graphics 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.*

### 5.7.1 Detailed Description

This file contains code used for graphics.

### 5.7.2 Macro Definition Documentation

#### 5.7.2.1 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.

### 5.7.3 Function Documentation

#### 5.7.3.1 Gclose()

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

Closes the graphics window.

##### Parameters

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

#### 5.7.3.2 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.
---------------	---

#### 5.7.3.3 Ginit()

```
void Ginit ( )
```

Initializes SDL2.

##### Warning

This function must be called before any other function.



#### 5.7.3.4 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.

#### 5.7.3.5 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.

#### 5.7.3.6 Gpre\_render\_cells()

```
SDL_Texture * Gpre_render_cells (
    Gwindow * window )
```

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 ( [Dither.h](#) ) to render the fading effect.

**5.7.3.7 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.

**5.7.3.8 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.
---------------	---

### 5.7.3.9 Gquit()

```
void Gquit ( )
```

Quits the SDL2 application.

#### Remarks

`exit()` should be called after this function.

### 5.7.3.10 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.

### 5.7.3.11 Gtextbox()

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

Creates a textbox in the graphics window.

#### Parameters

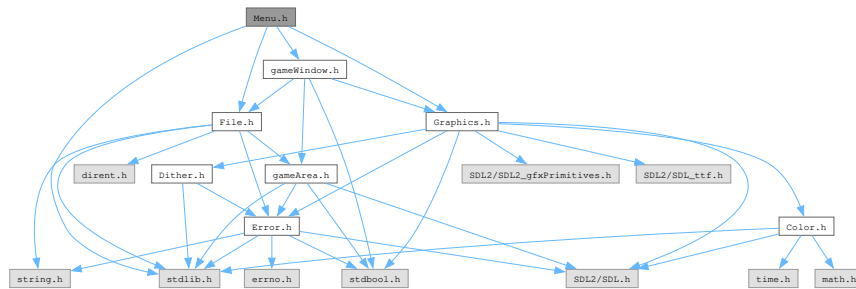
<i>window</i>	Pointer to the window to create textbox. Must not be NULL.
<i>text</i>	The text for the textbox. Empty string for no text. Must not be NULL.
<i>location</i>	The location for the textbox.
<i>col</i>	The color of the textbox, only accepts primary or secondary.
<i>border_width</i>	The width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

## 5.8 Menu.h File Reference

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

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

```
#include "Graphics.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`  
*Number of saves listed in the menu.*
- `#define INPUT_MAX_LENGTH 20`  
*Maximum length for the text inputs.*

## 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.*

## 5.8.1 Detailed Description

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

## 5.8.2 Function Documentation

### 5.8.2.1 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.

**5.8.2.2 Mclose()**

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

Closes the menu.

## Parameters

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

**5.8.2.3 Mevent()**

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

Handles an event in the menu.

## Parameters

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

## Remarks

It closes the program as intended if the user closes the window.

**5.8.2.4 Minit()**

```
Menu Minit ( )
```

Initializes the menu.

## Returns

The menu.



# Index

Aback  
    gameArea.h, [25](#)

Aclear  
    gameArea.h, [25](#)

Aflipcell  
    gameArea.h, [25](#)

Afree  
    gameArea.h, [25](#)

Agetage  
    gameArea.h, [26](#)

Anew  
    gameArea.h, [26](#)

Astep  
    gameArea.h, [26](#)

bg  
    Color\_theme, [7](#)

CELL\_SIZE  
    Graphics.h, [34](#)

Cinit  
    Color.h, [15](#)

Color.h, [13](#)  
    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, [16](#)

Dgenerate\_bayer\_matrix  
    Dither.h, [17](#)

Dither.h, [15](#)  
    Dfree\_bayer\_matrix, [16](#)  
    Dgenerate\_bayer\_matrix, [17](#)

Error.h, [17](#)  
    ErrorIFnoMemory, [19](#)  
    ErrorIFnull, [19](#)  
    ErrorIFsdl, [19](#)  
    ErrorIFtrue, [19](#)  
    ErrorIFtrue\_with\_params, [20](#)  
    ErrorWarning, [20](#)

ErrorIFnoMemory  
    Error.h, [19](#)

ErrorIFnull  
    Error.h, [19](#)

ErrorIFsdl  
    Error.h, [19](#)

ErrorIFtrue  
    Error.h, [19](#)

ErrorIFtrue\_with\_params  
    Error.h, [20](#)

ErrorWarning  
    Error.h, [20](#)

Fgame\_file, [8](#)

File.h, [20](#)  
    Flist, [22](#)  
    Fopen, [22](#)  
    Fsave, [22](#)

Flist  
    File.h, [22](#)

Fopen  
    File.h, [22](#)

Fsave  
    File.h, [22](#)

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

gameArea, [8](#)

gameArea.h, [23](#)  
    Aback, [25](#)  
    Aclear, [25](#)  
    Aflipcell, [25](#)  
    Afree, [25](#)  
    Agetage, [26](#)  
    Anew, [26](#)  
    Astep, [26](#)

gameWindow, [9](#)

gameWindow.h, [28](#)  
    Wclick, [29](#)  
    Wclose, [29](#)  
    Wdraw, [30](#)  
    Wevent, [30](#)  
    Winit, [30](#)  
    Wresetzoom, [30](#)  
    Wzoom, [32](#)

Gclose  
    Graphics.h, [34](#)

Gfill\_background

Graphics.h, [34](#)  
 Ginit  
     Graphics.h, [34](#)  
 Ginput\_text  
     Graphics.h, [34](#)  
 Gnew  
     Graphics.h, [35](#)  
 Gpre\_render\_cells  
     Graphics.h, [35](#)  
 Gprint  
     Graphics.h, [36](#)  
 Gprint\_title  
     Graphics.h, [36](#)  
 Gquit  
     Graphics.h, [36](#)  
 Graphics.h, [32](#)  
     CELL\_SIZE, [34](#)  
     Gclose, [34](#)  
     Gfill\_background, [34](#)  
     Ginit, [34](#)  
     Ginput\_text, [34](#)  
     Gnew, [35](#)  
     Gpre\_render\_cells, [35](#)  
     Gprint, [36](#)  
     Gprint\_title, [36](#)  
     Gquit, [36](#)  
     Gset\_color, [37](#)  
     Gtextbox, [37](#)  
 Gset\_color  
     Graphics.h, [37](#)  
 Gtextbox  
     Graphics.h, [37](#)  
 Gwindow, [10](#)  
  
 Mclick  
     Menu.h, [38](#)  
 Mclose  
     Menu.h, [39](#)  
 Menu, [11](#)  
 Menu.h, [37](#)  
     Mclick, [38](#)  
     Mclose, [39](#)  
     Mevent, [39](#)  
     Minit, [39](#)  
 Mevent  
     Menu.h, [39](#)  
 Minit  
     Menu.h, [39](#)  
  
 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, [29](#)  
 Wclose  
     gameWindow.h, [29](#)  
 Wdraw  
     gameWindow.h, [30](#)  
 Wevent  
     gameWindow.h, [30](#)  
 Winit  
     gameWindow.h, [30](#)  
 Wresetzoom  
     gameWindow.h, [30](#)  
 Wzoom  
     gameWindow.h, [32](#)