Simple language

Generated by Doxygen 1.9.1

Chapter 1

Game of life

Implementacja gry w życie Johna Conwaya

1.0.1 Kod źródłowy

https://github.com/mateuszkojro/ui4-game-of-life

1.0.2 Dokumentacja

https://mateuszkojro.github.io/ui4-game-of-life/

1.1 Wymagania

Aby skompilowac projekt wymagany jest przynajmniej standerd **C++ 14**

1.2 Budowa

Program sklada sie z:

- Interfejs prostego API silnika graficznego (`Renderer.h`) i jego implementacja w postacji prostego renderera do wyswietlania w konsoli (`SimpleConsoleRenderer.h`)
- Interfejs prostego API silnika gier (`GameEngine.h`) i jego implementacja w postaci tematycznej Gry w zycie (`GameOfLife.h`)

1.3 Zalozenia

W zalozeniu kazda gra stworzona z pomoca API `GameEngine` i `Renderer` powinna umozliwiac w prosty sposob zmiane silnika implementacje silnika graficznego moze to byc osiagniete implementujac wszystkie funckcje interfejsu `Renderer`. A nastepnie przekazujac wskaznik na instancje zaimplementowanego silnika do konfiguracji silnika gry np:

```
GameEngine::Config config;
config.renderer = new SimpleConsoleRenderer;
GameOfLife game_of_life(board, config);
```

gdzie `SimpleConsoleRenderer` to klasa dziedziczaca po `Renderer`

1.4 Mozliwa konfiguracja

Istnieje mozliwosc ustawienia poczatkowego stanu planszy za pomocą funkcjonalnosci udostepnionych przez klase `Board`:

1. Zapisywanie i odczytywanie stanu planszy z pliku:

```
    // mozemy otworzyc wczesniej zapisana plansze
const char[] PATH = "saved_board.data";
auto saved_board = Board::load_board(PATH);
// ustawiamy komurke na adresie x=1, y=1 jako aktywna
saved_board(1, 1) = true;
// Zapisujemy wprowadzone dane
Board::save_board(saved_board, "new_board.data")
```

1. Ustawienie zawartosci planszy za pomoca tablicy wartosci boolowskich gdzie `true` znaczy ze komurka bedzie zywa a `false` ze martwa

```
const size_x = 20, size_y = 20;
auto data = new bool[size_x * size_y];
memset(data, true, size_x * size_y);
data[11] = true;
data[12] = true;
data[13] = true;
Board board(data, size_x, size_y);
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

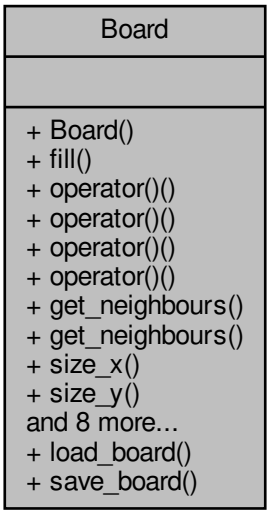
Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 Board Class Reference

`#include <Board.h>`
Collaboration diagram for Board:



Public Member Functions

- Board ()=delete
- void fill (bool value)
- bool & operator() (int x, int y)
- bool & operator() (int i)
- bool & operator() (int x, int y) const

  const qualified version operator()(int, int);
- bool & operator() (int i) const

  const qualified version operator()(int);
- std::array< bool, 9 > get_neighbours (int x, int y)
- std::array< bool, 9 > get_neighbours (int i)
- size_t size_x () const

- size_t size_y () const
- bool * get_board () const
- Board (bool *board, size_t x, size_t y)
- Board (size_t x, size_t y)
- Board (const Board &other)
- Board (Board &&other) noexcept
- Board & operator= (const Board &)
- unsigned size () const

  *Return the size of the underlying array (width * height)*
- virtual ~Board ()

Static Public Member Functions

- static Board load_board (const std::string &path)
- static void save_board (const Board &, const std::string &path)

5.1.1 Detailed Description

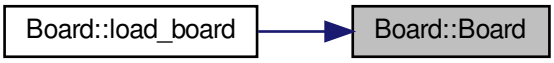
Class containing board (bool array) with functions useful for the implementation of the game of life
Definition at line 14 of file Board.h.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Board() [1/5]

```
Board::Board ( )  [delete]
```

Here is the caller graph for this function:



5.1.2.2 Board() [2/5]

```
Board::Board (
            bool * board,
            size_t x,
            size_t y )
```

Definition at line 9 of file Board.cc.

5.1.2.3 Board() [3/5]

```
Board::Board (
            size_t x,
            size_t y )
```

Definition at line 15 of file Board.cc.

5.1.2.4 Board() [4/5]

```
Board::Board (
            const Board & other )
```

Definition at line 105 of file Board.cc.

5.1.2.5 Board() [5/5]

```
Board::Board (
            Board && other )  [noexcept]
```

Definition at line 116 of file Board.cc.

5.1.2.6 ~Board()

```
Board::~Board ( )  [virtual]
```

Definition at line 19 of file Board.cc.

5.1.3 Member Function Documentation

5.1.3.1 fill()

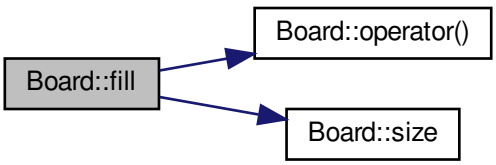
```
void Board::fill (
            bool value )
```

Fils the board wirh specified value

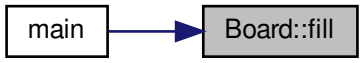
Parameters

value	to fill the buffer with

Definition at line 32 of file Board.cc.
Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.2 get_board()

```
bool * Board::get_board ( ) const
```

Get the ptr to bool array

Returns

ptr to bool array containing board data

Definition at line 38 of file Board.cc.

5.1.3.3 get_neighbours() [1/2]

```
std::array< bool, 9 > Board::get_neighbours (
            int i )
```

its an analog for get_neighbours(int x, int y); returns array of pairs to neighbours Some might be null

Parameters

i	address of the underlying array

Definition at line 46 of file Board.cc.

5.1.3.4 get_neighbours() [2/2]

```
std::array< bool, 9 > Board::get_neighbours (
            int x,
            int y )
```

Returns array of pairs to neighbours Some might be null

Parameters

x	x coordinate
y	y coordinate

Returns

array of pointers to neighbour cells (some might be null)

Definition at line 42 of file Board.cc.
Here is the caller graph for this function:

5.1.3.5 load_board()

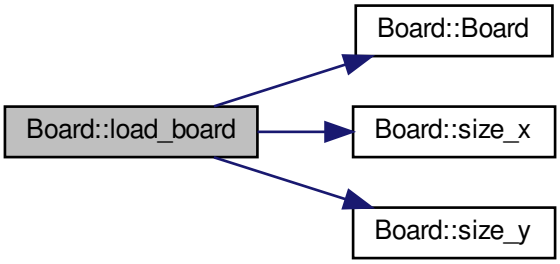
```
Board Board::load_board (
            const std::string & path )  [static]
```

Load board from the file

Parameters

Path	to the file containing the Board

Definition at line 82 of file Board.cc.
Here is the call graph for this function:



5.1.3.6 operator()() [1/4]

```
bool & Board::operator() (
            int i )
```

Accesses element of underlying arr

Parameters

i	Access the ith element of underlying arr

Returns

reference to given field

Definition at line 148 of file Board.cc.

5.1.3.7 operator()() [2/4]

```
bool & Board::operator() (
            int i ) const
```

const qualified version operator()(int);
Definition at line 156 of file Board.cc.

5.1.3.8 operator()() [3/4]

```
bool & Board::operator() (
            int x,
            int y )
```

Accesses element of underlying arr

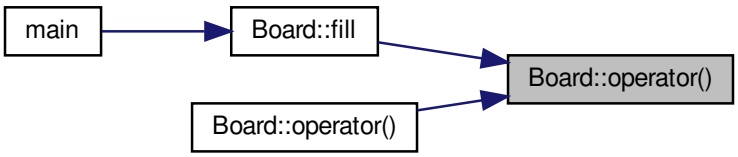
Parameters

x	x coordinate
y	y coordinate

Returns

reference to given field

Definition at line 152 of file Board.cc.
Here is the caller graph for this function:



5.1.3.9 operator()() [4/4]

```
bool & Board::operator() (
            int x,
            int y ) const
```

const qualified version operator()(int, int);
Definition at line 160 of file Board.cc.

Here is the call graph for this function:

5.1.3.10 operator=()

```
Board & Board::operator= (
            const Board & other )
```

Definition at line 124 of file Board.cc.

5.1.3.11 save_board()

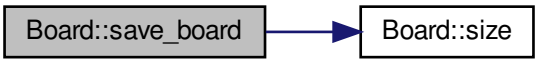
```
void Board::save_board (
            const Board & board,
            const std::string & path )  [static]
```

Save board state to the file

Parameters

Board	to be saved
Path	to the file that bord should be saved to default: "board.save"

Definition at line 95 of file Board.cc.
Here is the call graph for this function:

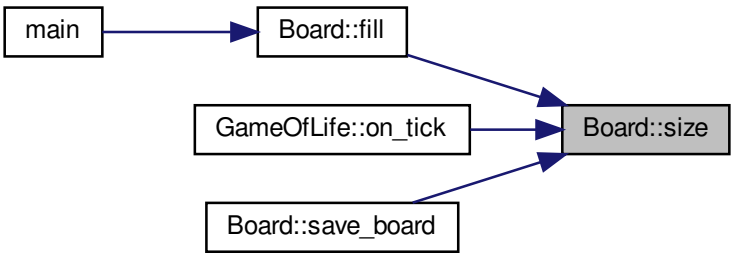


5.1.3.12 size()

```
unsigned Board::size ( ) const
```

Return the size of the underlying array (width ∗ height)
Definition at line 131 of file Board.cc.

Here is the caller graph for this function:



5.1.3.13 size_x()

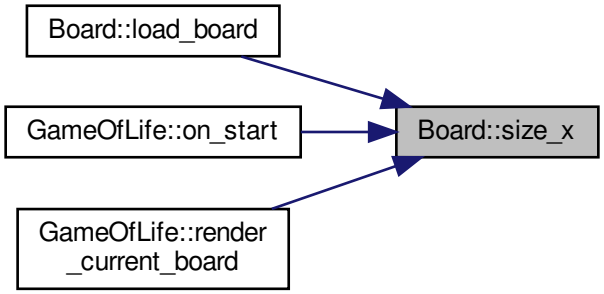
`size_t Board::size_x ( ) const`

Return size in x axis

Returns

width of the Board

Definition at line 24 of file Board.cc.
Here is the caller graph for this function:



5.1.3.14 size_y()

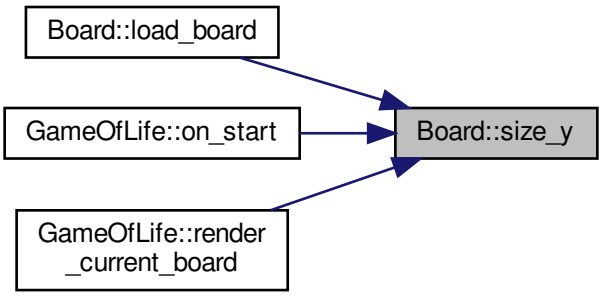
`size_t Board::size_y ( ) const`

Return size in y axis

Returns

height in y axis

Definition at line 28 of file Board.cc.
Here is the caller graph for this function:



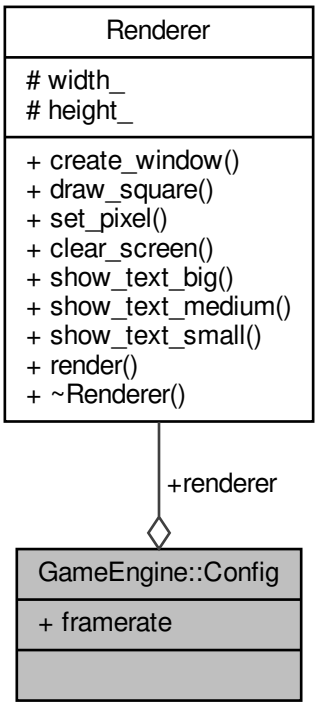
The documentation for this class was generated from the following files:

- Board.h
- Board.cc

5.2 GameEngine::Config Struct Reference

Config for game engines.
`#include <GameEngine.h>`

Collaboration diagram for GameEngine::Config:



Public Attributes

- int framerate
- Renderer * renderer

5.2.1 Detailed Description

Config for game engines.
Definition at line 25 of file GameEngine.h.

5.2.2 Member Data Documentation

5.2.2.1 framerate

```
int GameEngine::Config::framerate
```

Definition at line 26 of file GameEngine.h.

5.2.2.2 renderer

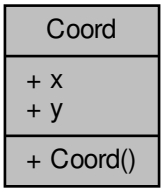
```
Renderer* GameEngine::Config::renderer
```

Definition at line 27 of file GameEngine.h.
The documentation for this struct was generated from the following file:

- GameEngine.h

5.3 Coord Struct Reference

Struct containing coordinates of different objects.
`#include <Renderer.h>`
Collaboration diagram for Coord:



Public Member Functions

- Coord (int x_in, int y_in)

Public Attributes

- int x
- int y

5.3.1 Detailed Description

Struct containing coordinates of different objects.
Definition at line 11 of file Renderer.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Coord()

```
Coord::Coord (
            int x_in,
            int y_in )  [inline]
```

Definition at line 12 of file Renderer.h.

5.3.3 Member Data Documentation

5.3.3.1 x

```
int Coord::x
```

Definition at line 15 of file Renderer.h.

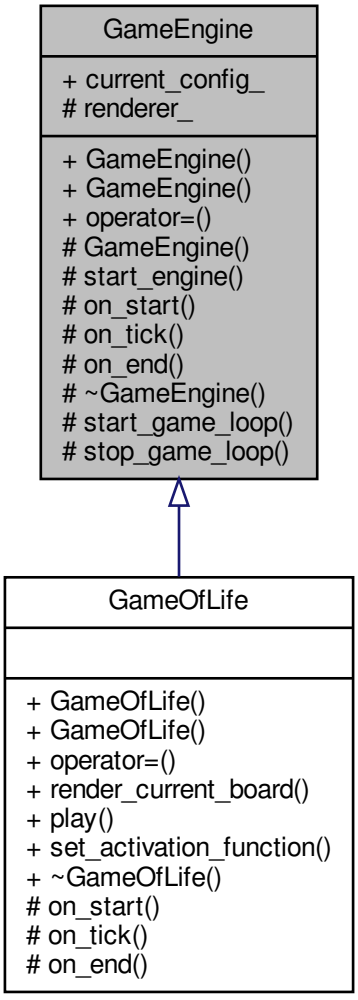
5.3.3.2 y

`int Coord::y`
Definition at line 16 of file Renderer.h.
The documentation for this struct was generated from the following file:

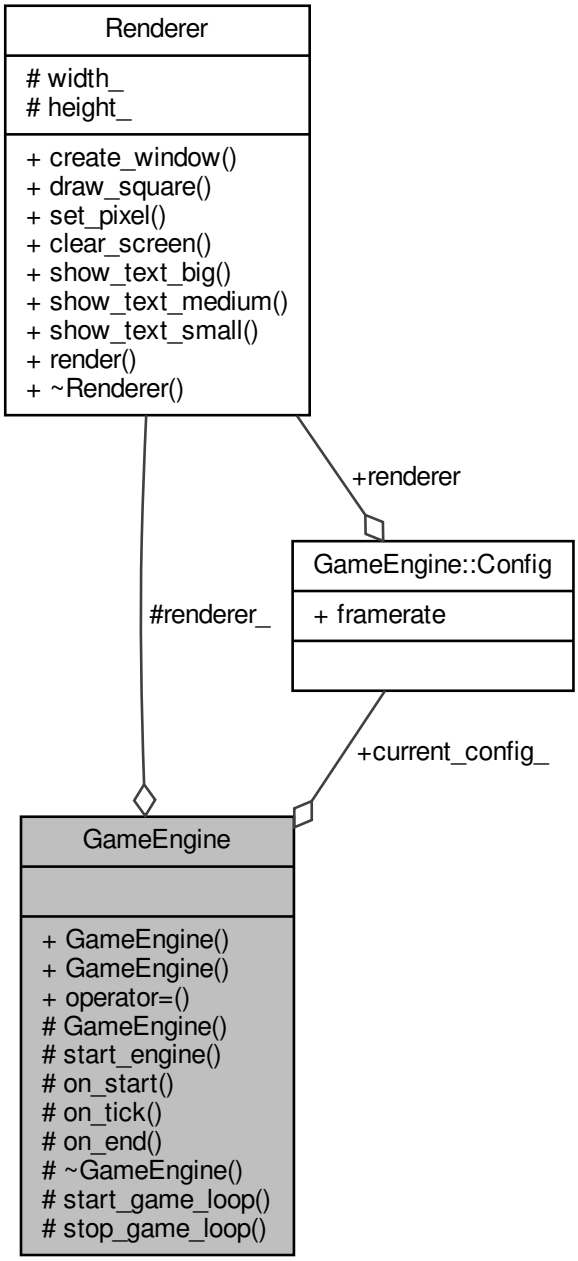
- Renderer.h

5.4 GameEngine Class Reference

Base class for custom game engines.
`#include <GameEngine.h>`
Inheritance diagram for GameEngine:



Collaboration diagram for GameEngine:



Classes

- struct Config

  Config for game engines.

Public Member Functions

- GameEngine ()=delete

We dont wanna allow creating GameEngine without configuration.

- GameEngine (const GameEngine &)=delete

  We dont wanna allow copying our game engine.

- GameEngine & operator= (const GameEngine &)=delete

Public Attributes

- struct GameEngine::Config current_config_

Protected Member Functions

- GameEngine (const GameEngine::Config &config)

  when we create a GameEngine we always need to give it a config

- virtual void start_engine () final

  game engine will start working

- virtual void on_start ()=0

  This function is called on game start should be overrided by the deriving class.

- virtual void on_tick ()=0

  This function will be invoked on every world tick should be ovverided by the derriving class.

- virtual void on_end ()=0

  This function is called when the game ends should be ovverided by the derriving class.

- virtual ~GameEngine ()
- void start_game_loop ()

  start main game loop - now every frame on_tick() will be called

- void stop_game_loop ()

  stopping the game loop - the on_end() will be called next

Protected Attributes

- Renderer * renderer_

5.4.1 Detailed Description

Base class for custom game engines.
Definition at line 13 of file GameEngine.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 GameEngine() [1/3]

```
GameEngine::GameEngine ( )  [delete]
```

We dont wanna allow creating GameEngine without configuration.

5.4.2.2 GameEngine() [2/3]

```
GameEngine::GameEngine (
            const GameEngine &  )  [delete]
```

We dont wanna allow copying our game engine.

5.4.2.3 GameEngine() [3/3]

```
GameEngine::GameEngine (
            const GameEngine::Config & config )  [inline], [explicit], [protected]
```

when we create a GameEngine we always need to give it a config
Definition at line 33 of file GameEngine.h.

5.4.2.4 ~GameEngine()

```
virtual GameEngine::~GameEngine ( )  [inline], [protected], [virtual]
```

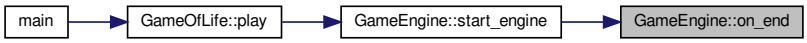
Definition at line 66 of file GameEngine.h.

5.4.3 Member Function Documentation

5.4.3.1 on_end()

```
virtual void GameEngine::on_end ( )  [protected], [pure virtual]
```

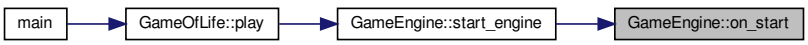
This function is called when the game ends should be ovverided by the derriving class.
Implemented in GameOfLife.
Here is the caller graph for this function:



5.4.3.2 on_start()

```
virtual void GameEngine::on_start ( )  [protected], [pure virtual]
```

This function is called on game start should be overrided by the deriving class.
Implemented in GameOfLife.
Here is the caller graph for this function:

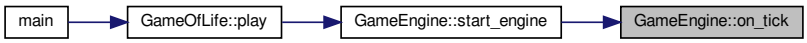


5.4.3.3 on_tick()

```
virtual void GameEngine::on_tick ( )  [protected], [pure virtual]
```

This function will be invoked on every world tick should be ovverided by the derriving class.
Implemented in GameOfLife.

Here is the caller graph for this function:



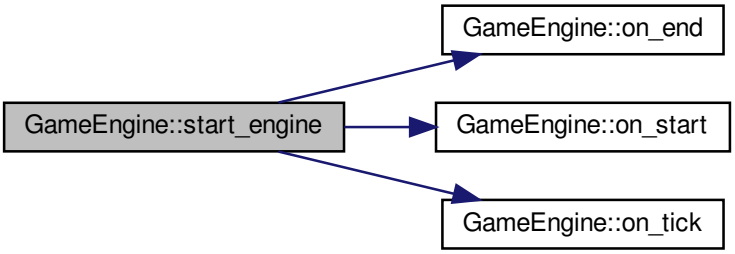
5.4.3.4 operator=()

```
GameEngine& GameEngine::operator= (
            const GameEngine &  )  [delete]
```

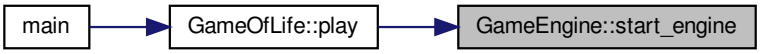
5.4.3.5 start_engine()

```
virtual void GameEngine::start_engine ( )  [inline], [final], [protected], [virtual]
```

game engine will start working
Definition at line 39 of file GameEngine.h.
Here is the call graph for this function:



Here is the caller graph for this function:

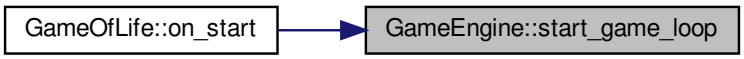


5.4.3.6 start_game_loop()

```
void GameEngine::start_game_loop ( )  [inline], [protected]
```

start main game loop - now every frame on_tick() will be called
Definition at line 70 of file GameEngine.h.

Here is the caller graph for this function:



5.4.3.7 stop_game_loop()

```
void GameEngine::stop_game_loop ( )  [inline], [protected]
```

stopping the game loop - the on_end() will be called next
Definition at line 73 of file GameEngine.h.

5.4.4 Member Data Documentation

5.4.4.1 current_config_

```
struct GameEngine::Config GameEngine::current_config_
```

5.4.4.2 renderer_

```
Renderer* GameEngine::renderer_  [protected]
```

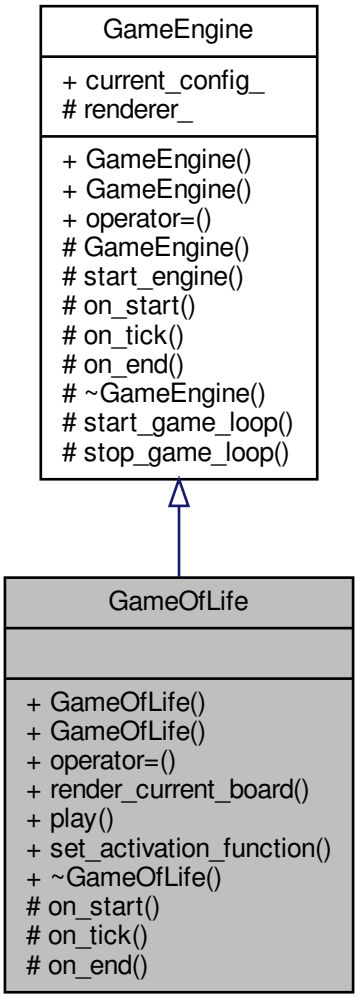
Definition at line 75 of file GameEngine.h.
The documentation for this class was generated from the following file:

- GameEngine.h

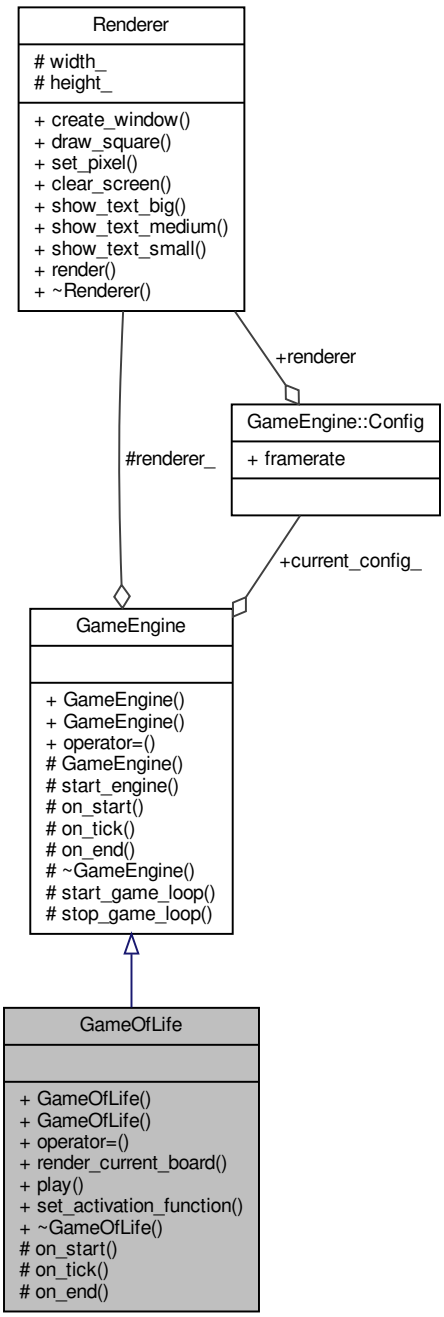
5.5 GameOfLife Class Reference

Implementation of the game of life.
`#include <GameOfLife.h>`

Inheritance diagram for GameOfLife:



Collaboration diagram for GameOfLife:



Public Member Functions

- GameOfLife (const Board &board, const Config &config)
- GameOfLife (const GameOfLife &)=delete
- const GameOfLife & operator= (const GameOfLife &)=delete
- void render_current_board ()

  *render current_board_*
- void play ()

start the game engine

- void set_activation_function (bool(∗func)(bool, int))
- ∼GameOfLife () override

Protected Member Functions

- void on_start () override

  This function is called on game start should be overrided by the deriving class.
- void on_tick () override

  This function will be invoked on every world tick should be ovverided by the derriving class.
- void on_end () override

  This function is called when the game ends should be ovverided by the derriving class.

Additional Inherited Members

5.5.1 Detailed Description

Implementation of the game of life.
Definition at line 18 of file GameOfLife.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 GameOfLife() [1/2]

```
GameOfLife::GameOfLife (
            const Board & board,
            const Config & config )  [inline], [explicit]
```

Definition at line 20 of file GameOfLife.h.

5.5.2.2 GameOfLife() [2/2]

```
GameOfLife::GameOfLife (
            const GameOfLife &  )  [delete]
```

5.5.2.3 ∼GameOfLife()

```
GameOfLife::∼GameOfLife ( )  [override]
```

sets the function that will be used to determine if cell should be alive

Parameters

func	returning bool (if is alive) has params (bool) is currently alive (int) how many neighbours it has

Definition at line 26 of file GameOfLife.cc.

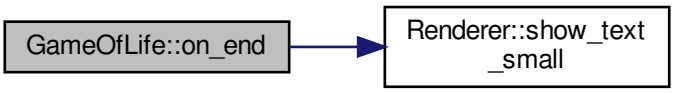
5.5.3 Member Function Documentation

5.5.3.1 on_end()

```
void GameOfLife::on_end ( )  [override], [protected], [virtual]
```

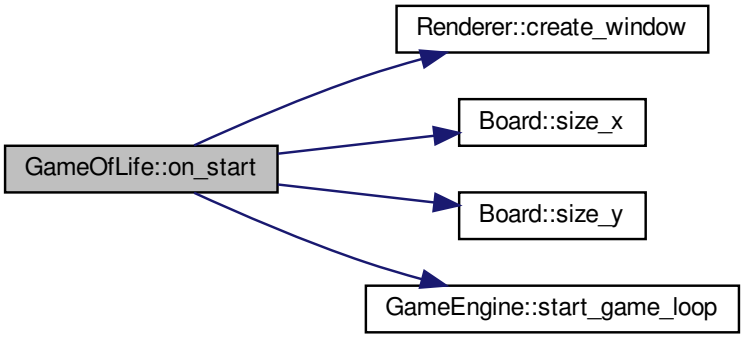
This function is called when the game ends should be ovverided by the derriving class.
Implements GameEngine.

Definition at line 103 of file GameOfLife.cc.
Here is the call graph for this function:



5.5.3.2 on_start()

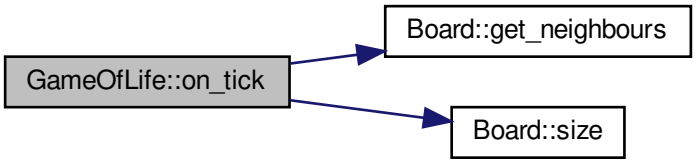
`void GameOfLife::on_start ( ) [override], [protected], [virtual]`
This function is called on game start should be overrided by the deriving class.
Implements GameEngine.
Definition at line 33 of file GameOfLife.cc.
Here is the call graph for this function:



5.5.3.3 on_tick()

`void GameOfLife::on_tick ( ) [override], [protected], [virtual]`
This function will be invoked on every world tick should be ovverided by the derriving class.
Implements GameEngine.
Definition at line 65 of file GameOfLife.cc.

Here is the call graph for this function:



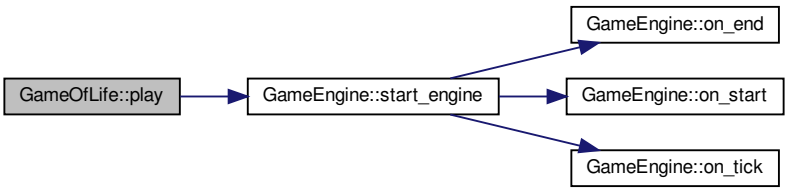
5.5.3.4 operator=()

```
const GameOfLife& GameOfLife::operator= (
            const GameOfLife &  )  [delete]
```

5.5.3.5 play()

```
void GameOfLife::play ( )
```

start the game engine
Definition at line 18 of file GameOfLife.cc.
Here is the call graph for this function:

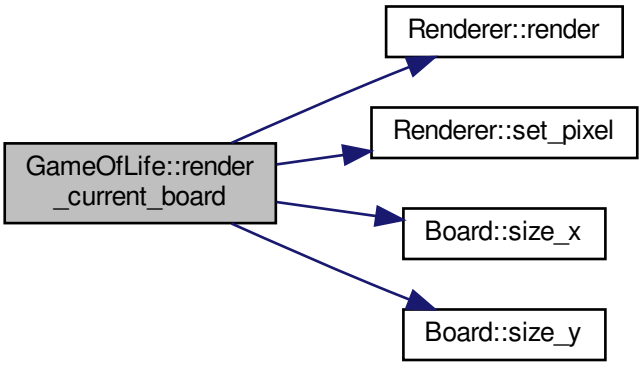


Here is the caller graph for this function:

5.5.3.6 render_current_board()

```
void GameOfLife::render_current_board ( )
```

render current_board_
Definition at line 50 of file GameOfLife.cc.
Here is the call graph for this function:



5.5.3.7 set_activation_function()

```
void GameOfLife::set_activation_function (
            bool(*)(bool, int) func )
```

Sets the function that will be used to determine if the cell should be alive

Parameters

func	function ptr function should return bool (true if a cell should be alive) based on the number of alive neighbours and if given cell is alive

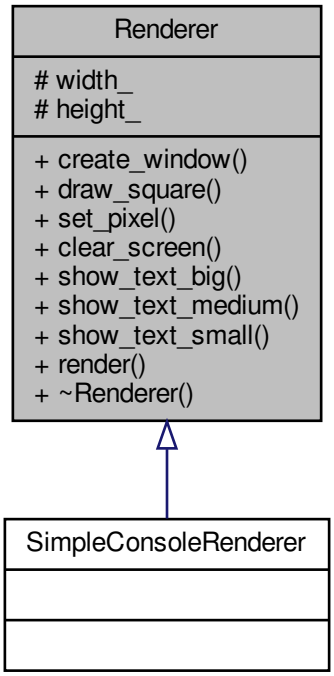
Definition at line 22 of file GameOfLife.cc.
The documentation for this class was generated from the following files:

- GameOfLife.h
- GameOfLife.cc

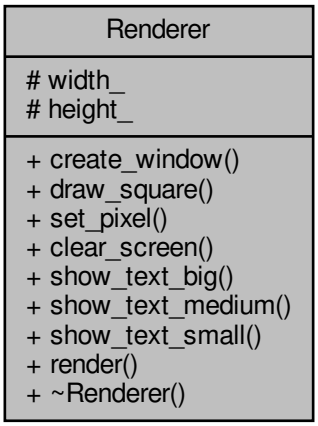
5.6 Renderer Class Reference

Basic base class for all renderers.
`#include <Renderer.h>`

Inheritance diagram for Renderer:



Collaboration diagram for Renderer:



Public Member Functions

- virtual void create_window (int size_x, int size_y)=0

Creates window of given size.

- virtual void draw_square (const Coord &position, int size_x, int size_y, const Color &fill)=0

  Draws a square on position wit size and fill.
- virtual void set_pixel (const Coord &position, const Color &fill)=0
- virtual void clear_screen (const Color &fill)=0

  Fill all screen with defined color.
- virtual void show_text_big (const Coord &position, const std::string &text)=0
- virtual void show_text_medium (const Coord &position, const std::string &text)=0
- virtual void show_text_small (const Coord &position, const std::string &text)=0
- virtual void render ()=0
- virtual ~Renderer ()=default

Protected Attributes

- int width_

  width of the render plane
- int height_

  height of the render plane

5.6.1 Detailed Description

Basic base class for all renderers.
Definition at line 29 of file Renderer.h.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ~Renderer()

```
virtual Renderer::~Renderer ( )  [virtual], [default]
```

5.6.3 Member Function Documentation

5.6.3.1 clear_screen()

```
virtual void Renderer::clear_screen (
            const Color & fill )  [pure virtual]
```

Fill all screen with defined color.

Parameters

Color	color to fill the screen with

5.6.3.2 create_window()

```
virtual void Renderer::create_window (
            int size_x,
            int size_y )  [pure virtual]
```

Creates window of given size.

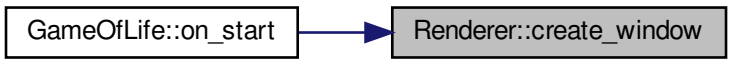
Parameters

int	Size in x dimension

Parameters

int	Size in y dimension

Here is the caller graph for this function:



5.6.3.3 draw_square()

```
virtual void Renderer::draw_square (
            const Coord & position,
            int size_x,
            int size_y,
            const Color & fill )  [pure virtual]
```

Draws a square on position wit size and fill.

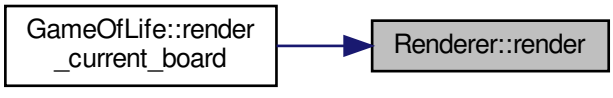
Parameters

Coord	position
size	in x axis
size	in y axis
Color	color to fill square with

5.6.3.4 render()

```
virtual void Renderer::render ( )  [pure virtual]
```

Here is the caller graph for this function:



5.6.3.5 set_pixel()

```
virtual void Renderer::set_pixel (
```

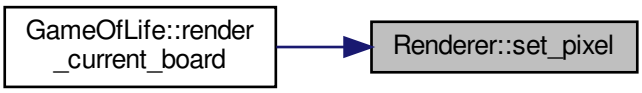
```
            const Coord & position,
            const Color & fill )  [pure virtual]
```

Sets the position at given coordinate to given Color

Parameters

position	const Coord& position of the pixels to be set
fill	The color to set the pixel to

Here is the caller graph for this function:



5.6.3.6 show_text_big()

```
virtual void Renderer::show_text_big (
            const Coord & position,
            const std::string & text )  [pure virtual]
```

Show text in big letters on position

Parameters

position	Coord of the beginning of the text
text	text to be printed

5.6.3.7 show_text_medium()

```
virtual void Renderer::show_text_medium (
            const Coord & position,
            const std::string & text )  [pure virtual]
```

Show text in medium letters on position

Parameters

position	Coord of the beginning of the text
text	text to be printed

5.6.3.8 show_text_small()

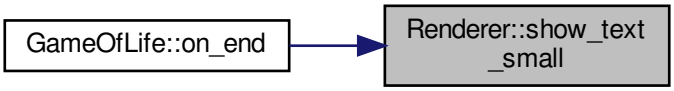
```
virtual void Renderer::show_text_small (
            const Coord & position,
            const std::string & text )  [pure virtual]
```

Show text in small letters on position

Parameters

position	Coord of the beginning of the text
text	text to be printed

Here is the caller graph for this function:



5.6.4 Member Data Documentation

5.6.4.1 height_

`int Renderer::height_ [protected]`
height of the render plane
Definition at line 76 of file Renderer.h.

5.6.4.2 width_

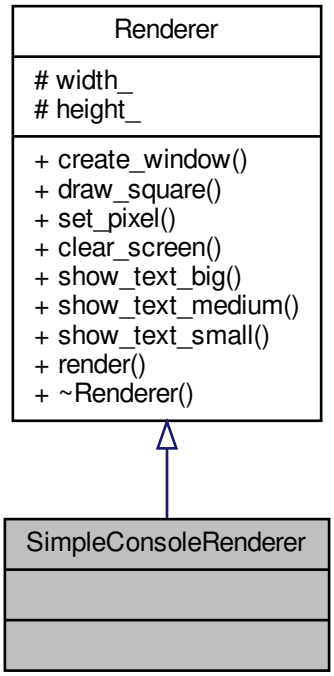
`int Renderer::width_ [protected]`
width of the render plane
Definition at line 74 of file Renderer.h.
The documentation for this class was generated from the following file:

- Renderer.h

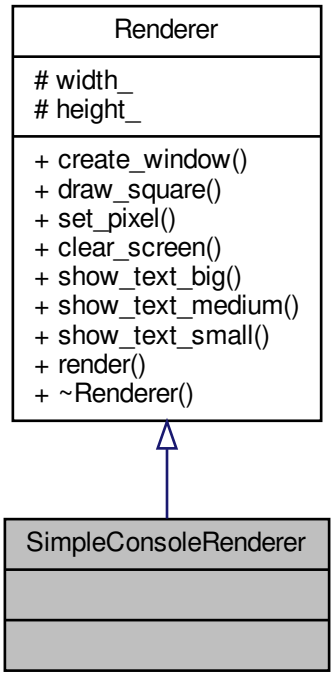
5.7 SimpleConsoleRenderer Class Reference

`#include <SimpleConsoleRenderer.h>`

Inheritance diagram for SimpleConsoleRenderer:



Collaboration diagram for SimpleConsoleRenderer:



Additional Inherited Members

5.7.1 Detailed Description

Definition at line 12 of file SimpleConsoleRenderer.h.
The documentation for this class was generated from the following files:

- SimpleConsoleRenderer.h
- SimpleConsoleRenderer.cc

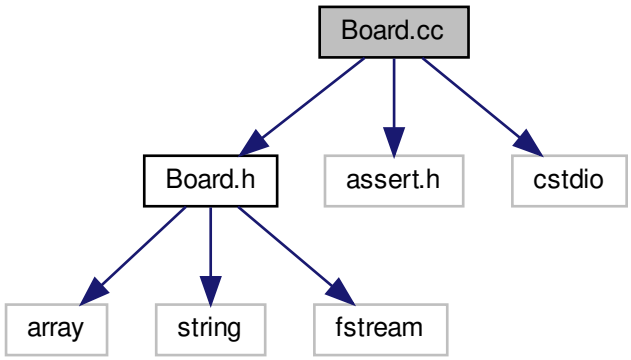
Chapter 6

File Documentation

6.1 Board.cc File Reference

```
#include "Board.h"
#include <assert.h>
#include <cstdio>
```

Include dependency graph for Board.cc:



6.2 Board.cc

```
00001 //
00002 // Created by mateu on 4/1/2021.
00003 //
00004
00005 #include "Board.h"
00006 #include <assert.h>
00007 #include <cstdio>
00008
00009 Board::Board(bool *board, size_t x, size_t y) :
00010         size_x_(x),
00011         size_y_(y) {
00012     board_ = board;
00013 }
00014
00015 Board::Board(size_t x, size_t y) : size_x_(x), size_y_(y) {
00016   board_ = new bool[x * y];
00017 }
00018
00019 Board::~Board() {
00020     delete[] board_;
00021     board_ = nullptr;
```

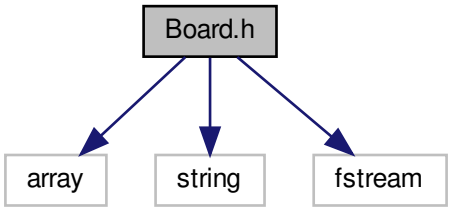
```
00022 }
00023
00024 size_t Board::size_x() const {
00025     return size_x_;
00026 }
00027
00028 size_t Board::size_y() const {
00029     return size_y_;
00030 }
00031
00032 void Board::fill(bool value) {
00033   for (int i = 0; i < size(); i++) {
00034     operator()(i) = value;
00035   }
00036 }
00037
00038 bool *Board::get_board() const {
00039     return board_;
00040 }
00041
00042 std::array<bool, 9> Board::get_neighbours(int x, int y) {
00043     return get_neighbours(translate_adress(x, y));
00044 }
00045
00046 std::array<bool, 9> Board::get_neighbours(int i) {
00047     std::array<bool, 9> result{};
00048
00049
00050     int pos_x, pos_y;
00051
00052     // Transform continuous address to x, y one
00053     pos_y = i / size_x_;
00054     pos_x = i % size_x_;
00055
00056     // place in out array
00057     int itr = 0;
00058
00059
00060     for (int y = -1; y <= 1; y++) {
00061         for (int x = -1; x <= 1; x++) {
00062
00063             // transform to position on the board
00064             int board_x = pos_x + x;
00065             int board_y = pos_y + y;
00066             int board_i = translate_adress(board_x, board_y);
00067
00068             bool is_target = (board_x == pos_x) && (board_y == pos_y);
00069             bool is_valid = (board_i != -1);
00070
00071             if (!is_target && is_valid){
00072                 result[itr++] = board_[board_i];
00073             } else {
00074                 result[itr++] = false;
00075             }
00076         }
00077     }
00078
00079     return result;
00080 }
00081
00082 Board Board::load_board(const std::string &path) {
00083     std::fstream file;
00084     file.open(path, std::ios::in);
00085     size_t size_x, size_y;
00086
00087     file » size_x » size_y;
00088     bool *board = new bool[size_y * size_x];
00089     for (int i = 0; i < size_x * size_y; ++i) {
00090         file » board[i];
00091     }
00092     return Board(board, size_x, size_y);
00093 }
00094
00095 void Board::save_board(const Board &board, const std::string &path) {
00096     std::fstream file;
00097     file.open(path, std::ios::out);
00098     file « board.size_x_;
00099     file « board.size_y_;
00100     for (int i = 0; i < board.size(); ++i) {
00101         file « board(i);
00102     }
00103 }
00104
00105 Board::Board(const Board &other) {
00106     copy(other);
00107 }
00108
```

```
00109 void Board::copy(const Board &other) {
00110     size_y_ = other.size_y_;
00111     size_x_ = other.size_x_;
00112     board_ = new bool[size()];
00113     memcpy(board_, other.board_, size());
00114 }
00115
00116 Board::Board(Board &&other) noexcept {
00117     assert(false);
00118     size_y_ = other.size_y_;
00119     size_x_ = other.size_x_;
00120     board_ = other.board_;
00121     other.board_ = nullptr;
00122 }
00123
00124 Board &Board::operator=(const Board &other) {
00125     if (this == &other)
00126         return *this;
00127     copy(other);
00128     return *this;
00129 }
00130
00131 unsigned Board::size() const {
00132     return size_x_ * size_y_;
00133 }
00134
00135 int Board::translate_adress(int x, int y) const {
00136   if (y >= size_y_)
00137     return -1;
00138   if (x >= size_x_)
00139     return -1;
00140   if (x < 0)
00141     return -1;
00142   if (y < 0)
00143     return -1;
00144
00145   return y * size_x_ + x;
00146 }
00147
00148 bool &Board::operator()(int i) {
00149     return board_[i];
00150 }
00151
00152 bool &Board::operator()(int x, int y) {
00153     return operator()(translate_adress(x, y));
00154 }
00155
00156 bool &Board::operator()(int i) const {
00157     return board_[i];
00158 }
00159
00160 bool &Board::operator()(int x, int y) const {
00161     return operator()(translate_adress(x, y));
00162 }
```

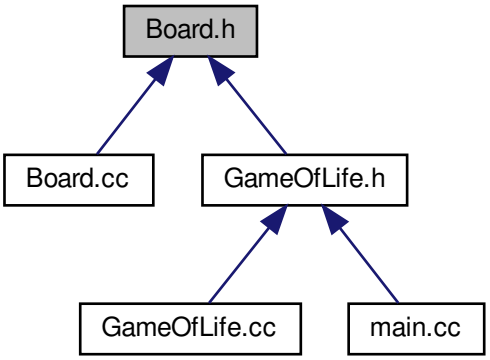
6.3 Board.h File Reference

```
#include <array>
#include <string>
#include <fstream>
```

Include dependency graph for Board.h:



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



Classes

- class Board

6.4 Board.h

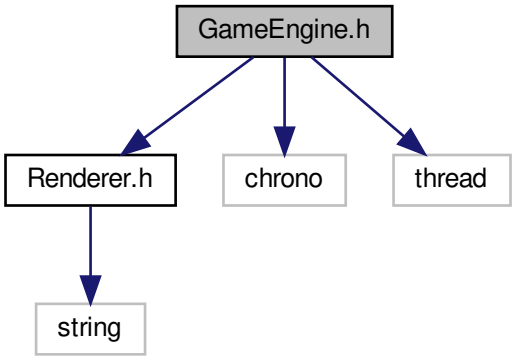
```
00001 //
00002 // Created by mateu on 4/1/2021.
00003 //
00004
00005 #ifndef GAME_OF_LIFE_BOARD_H
00006 #define GAME_OF_LIFE_BOARD_H
00007
00008 #include <array>
00009 #include <string>
00010 #include <fstream>
00011
00014 class Board {
00015 public:
00016
00017     Board() = delete;
00018
00021     void fill(bool value);
00022
00027     bool &operator()(int x, int y);
```

```
00028
00032     bool &operator()(int i);
00033
00035     bool &operator()(int x, int y) const;
00036
00038     bool &operator()(int i) const;
00039
00044     std::array<bool, 9> get_neighbours(int x, int y);
00045
00049     std::array<bool, 9> get_neighbours(int i);
00050
00053     static Board load_board(const std::string &path);
00054
00058     static void save_board(const Board &, const std::string &path);
00059
00062     size_t size_x() const;
00063
00066     size_t size_y() const;
00067
00070     bool *get_board() const;
00071
00072 public:
00073
00074
00075     Board(bool *board, size_t x, size_t y);
00076
00077     Board(size_t x, size_t y);
00078
00079     Board(const Board &other);
00080
00081     Board(Board &&other) noexcept;
00082
00083     Board &operator=(const Board &);
00084
00086     unsigned size() const;
00087
00088     virtual ~Board();
00089
00090
00091 private:
00096     int translate_adress(int x, int y) const;
00097
00100     void copy(const Board &other);
00101
00102     size_t size_x_;
00103     size_t size_y_;
00104
00106     bool *board_;
00107
00108 };
00109
00110
00111 #endif //GAME_OF_LIFE_BOARD_H
```

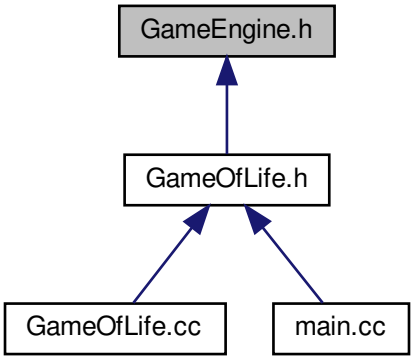
6.5 GameEngine.h File Reference

```
#include "Renderer.h"
#include <chrono>
#include <thread>
```

Include dependency graph for GameEngine.h:



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



Classes

- class GameEngine

  Base class for custom game engines.
- struct GameEngine::Config

  Config for game engines.

6.6 GameEngine.h

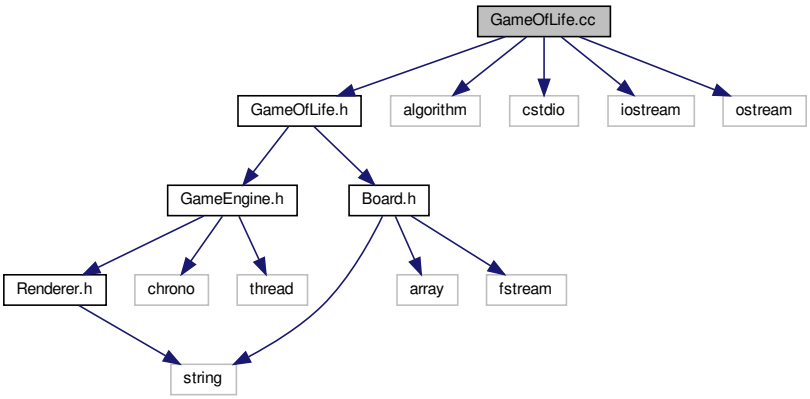
```
00001 //
00002 // Created by mateu on 3/22/2021.
00003 //
00004
00005 #ifndef GAME_OF_LIFE_GAMEENGINE_H
00006 #define GAME_OF_LIFE_GAMEENGINE_H
00007
```

```
00008 #include "Renderer.h"
00009 #include <chrono>
00010 #include <thread>
00011
00013 class GameEngine {
00014
00015 public:
00017     GameEngine() = delete;
00018
00020     GameEngine(const GameEngine &) = delete;
00021
00022     GameEngine &operator=(const GameEngine &) = delete;
00023
00025     struct Config {
00026         int framerate;
00027         Renderer *renderer;
00028     } current_config_;
00029
00030 protected:
00031
00033     explicit GameEngine(const GameEngine::Config &config) :
00034             current_config_(config),
00035             renderer_(config.renderer),
00036             running_(false) {};
00037
00039     virtual void start_engine() final {
00040         on_start();
00041         while (running_) {
00042             // we are measuring the time before the work in the frame
00043             auto start = std::chrono::high_resolution_clock::now();
00044             on_tick();
00045             // we are measuring the time after the work in the frame
00046             auto stop = std::chrono::high_resolution_clock::now();
00047             // how long should the frame take
00048             auto target_frame_time = std::chrono::seconds(1 / current_config_.framerate);
00049             // how long it took
00050             auto current_frame_time = start - stop;
00051             // sleep for the difference between target and real time
00052             std::this_thread::sleep_for(target_frame_time - current_frame_time);
00053         }
00054         on_end();
00055     };
00056
00058     virtual void on_start() = 0;
00059
00061     virtual void on_tick() = 0;
00062
00064     virtual void on_end() = 0;
00065
00066     virtual ~GameEngine() { delete renderer_; }
00067
00068 protected:
00070     void start_game_loop() { running_ = true; }
00071
00073     void stop_game_loop() { running_ = false; }
00074
00075     Renderer *renderer_;
00076
00077 private:
00078     bool running_;
00079
00080 };
00081
00082
00083 #endif //GAME_OF_LIFE_GAMEENGINE_H
```

6.7 GameOfLife.cc File Reference

```
#include "GameOfLife.h"
#include <algorithm>
#include <cstdio>
#include <iostream>
#include <ostream>
```

Include dependency graph for GameOfLife.cc:



Functions

- bool conway_activation (bool is_alive, int no_neighbours)

6.7.1 Function Documentation

6.7.1.1 conway_activation()

```
bool conway_activation (
            bool is_alive,
            int no_neighbours )
```

Activation function proposed by Conway in his original game

Parameters

is_alive	is cell that is checked alive
no_neighbours	how many neighbours are alive

Returns

should the cell be alive or not

Definition at line 112 of file GameOfLife.cc.

6.8 GameOfLife.cc

```
00001 //
00002 // Created by mateu on 3/24/2021.
00003 //
00004
00005 #include "GameOfLife.h"
00006 #include <algorithm>
00007 #include <cstdio>
00008 #include <iostream>
00009 #include <ostream>
00010 #include <cstdio>
00011
00012 static void show_board(const Board &board) {
00013   for (int i = 0; i < board.size(); i++) {
00014     std::cout « board(i) « " ";
```

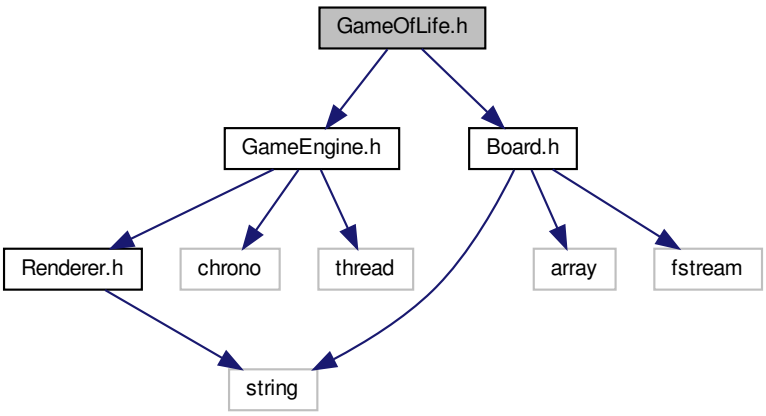
```
00015   }
00016 }
00017
00018 void GameOfLife::play() {
00019     start_engine();
00020 }
00021
00022 void GameOfLife::set_activation_function(bool (*func)(bool, int)) {
00023     activation_func_ = func;
00024 }
00025
00026 GameOfLife::~GameOfLife() {
00027     delete current_board_;
00028     delete next_board_;
00029     current_board_ = nullptr;
00030     next_board_ = nullptr;
00031 }
00032
00033 void GameOfLife::on_start() {
00034     // Show a welcome screen and ready to go
00035     renderer_->create_window(current_board_->size_x(),
00036                              current_board_->size_y());
00037
00038 //    renderer_->clear_screen(Color::Black);
00039 //    renderer_->show_text_big(Coord{0, 0}, "Gra");
00040 //    renderer_->show_text_big(Coord{1, 0}, "w");
00041 //    renderer_->show_text_big(Coord{2, 0}, "zycie");
00042 //    renderer_->render();
00043 //    std::string text;
00044 //    while (text != "start") {
00045 //        std::cin » text;
00046 //    }
00047     start_game_loop();
00048 }
00049
00050 void GameOfLife::render_current_board() {
00051     for (int x = 0; x < current_board_->size_x(); x++) {
00052         for (int y = 0; y < current_board_->size_y(); y++) {
00053             Color color;
00054             if ((*current_board_)(x, y))
00055                 color = Color::White;
00056             else
00057                 color = Color::Black;
00058
00059             renderer_->set_pixel(Coord(x, y), color);
00060         }
00061     }
00062     renderer_->render();
00063 }
00064
00065 void GameOfLife::on_tick() {
00066   // loop through all the active and inactive cells
00067   for (int i = 0; i < current_board_->size(); ++i) {
00068     // get neighbours for current cell
00069     auto neighbours = current_board_->get_neighbours(i);
00070     // make a cell alive if activation function determines so
00071     bool value = activation_func_(
00072         // Whats the current state of the cell
00073         (*next_board_)(i),
00074         // Count cells that are active around this cell
00075         std::count(neighbours.begin(), neighbours.end(), true));
00076
00077 #if DEBUG
00078     std::printf("Value at cell %d is %d input was (%lld)\n", i, value,
00079                 std::count(neighbours.begin(), neighbours.end(), true));
00080 #endif
00081
00082     (*next_board_)(i) = value;
00083   }
00084
00085 #if DEBUG
00086   static int tick = 0;
00087   std::printf("Tick %d\n", tick++);
00088   std::printf("Current board\n");
00089   show_board(*current_board_);
00090   std::printf("\nNext board\n");
00091   show_board(*next_board_);
00092   std::printf("\n\n");
00093 #else
00094   render_current_board();
00095 #endif
00096
00097   Board temp = *next_board_;
00098   *next_board_ = *current_board_;
00099   *current_board_ = temp;
00100
00101 }
```

```
00102
00103 void GameOfLife::on_end() {
00104     // Show some stats on exit
00105     renderer_->show_text_small(Coord(0, 0), "Dziekuje!");
00106 }
00107
00112 bool conway_activation(bool is_alive, int no_neighbours) {
00113     if (is_alive) {
00114         if (no_neighbours == 2 || no_neighbours == 3)
00115             return true;
00116         else
00117             return false;
00118     } else {
00119         if (no_neighbours == 3)
00120             return true;
00121         else
00122             return false;
00123     }
00124 }
```

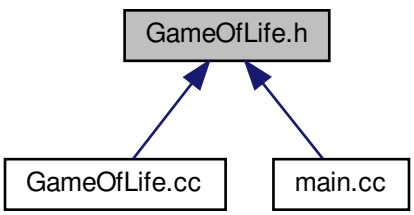
6.9 GameOfLife.h File Reference

```
#include "GameEngine.h"
#include "Board.h"
```

Include dependency graph for GameOfLife.h:



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



Classes

- class GameOfLife

  Implementation of the game of life.

Functions

- bool conway_activation (bool is_alive, int no_neighbours)

6.9.1 Function Documentation

6.9.1.1 conway_activation()

```
bool conway_activation (
            bool is_alive,
            int no_neighbours )
```

Activation function based on the original Conway's Game of Life

Parameters

is_alive	is the cell alive or not
no_neighbours	how many alive neighbours are around

Returns

true if cell should be alive false if not

Activation function proposed by Conway in his original game

Parameters

is_alive	is cell that is checked alive
no_neighbours	how many neighbours are alive

Returns

should the cell be alive or not

Definition at line 112 of file GameOfLife.cc.

6.10 GameOfLife.h

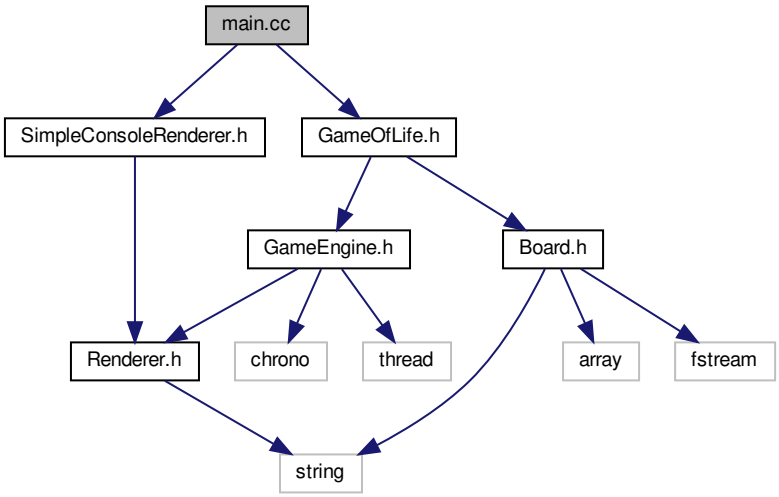
```
00001 //
00002 // Created by mateu on 3/24/2021.
00003 //
00004
00005 #ifndef GAME_OF_LIFE_GAMEOFLIFE_H
00006 #define GAME_OF_LIFE_GAMEOFLIFE_H
00007
00008 #include "GameEngine.h"
00009 #include "Board.h"
00010
00015 bool conway_activation(bool is_alive, int no_neighbours);
00016
00018 class GameOfLife : public GameEngine {
00019 public:
00020     explicit GameOfLife(const Board &board, const Config &config)
00021             : GameEngine(config),
00022               activation_func_(conway_activation) {
00023
00024         current_board_ = new Board(board);
00025         next_board_ = new Board(board);
```

```
00026     };
00027
00028
00029     GameOfLife(const GameOfLife &) = delete;
00030
00031     const GameOfLife &operator=(const GameOfLife &) = delete;
00032
00034     void render_current_board();
00035
00037     void play();
00038
00042     void set_activation_function(bool (*func)(bool, int));
00043
00047     ~GameOfLife() override;
00048
00049 protected:
00050
00051
00052     void on_start() override;
00053
00054     void on_tick() override;
00055
00056     void on_end() override;
00057
00058 private:
00059
00061     Board *current_board_;
00063     Board *next_board_;
00064
00066     bool (*activation_func_)(bool, int);
00067
00068 };
00069
00070 #endif //GAME_OF_LIFE_GAMEOFLIFE_H
```

6.11 main.cc File Reference

```
#include "SimpleConsoleRenderer.h"
#include "GameOfLife.h"
```

Include dependency graph for main.cc:



Functions

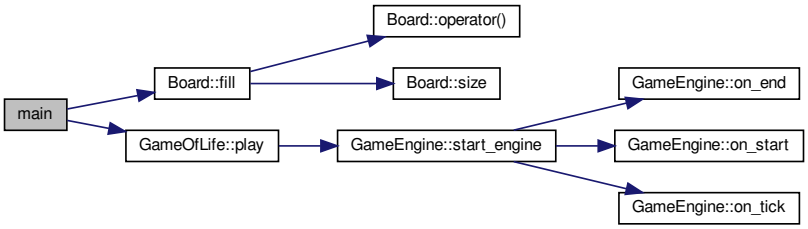
- int main ()

6.11.1 Function Documentation

6.11.1.1 main()

```
int main ( )
```

Definition at line 4 of file main.cc.
Here is the call graph for this function:



6.12 main.cc

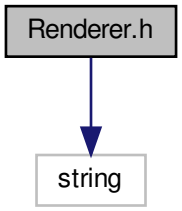
```
00001 #include "SimpleConsoleRenderer.h"
00002 #include "GameOfLife.h"
00003
00004 int main() {
00005     const unsigned size_x = 10;
00006     const unsigned size_y = 10;
00007
00008     GameEngine::Config config{};
00009
00010     config.framerate = 1;
00011     config.renderer = new SimpleConsoleRenderer;
00012
00013     Board board(size_x, size_y);
00014
00015     // Make the board empty
00016     board.fill(false);
00017
00018     // Setup some config on the board
00019     board(1, 1) = true;
00020     board(1, 2) = true;
00021     board(1, 3) = true;
00022
00023     board(3, 1) = true;
00024     board(3, 2) = true;
00025     board(3, 3) = true;
00026
00027     board(6, 1) = true;
00028     board(6, 2) = true;
00029     board(6, 3) = true;
00030
00031     board(9, 3) = true;
00032     board(9, 4) = true;
00033     board(9, 5) = true;
00034
00035     // Start the GameEngine
00036     GameOfLife game_of_life(board, config);
00037     game_of_life.play();
00038 }
```

6.13 README.md File Reference

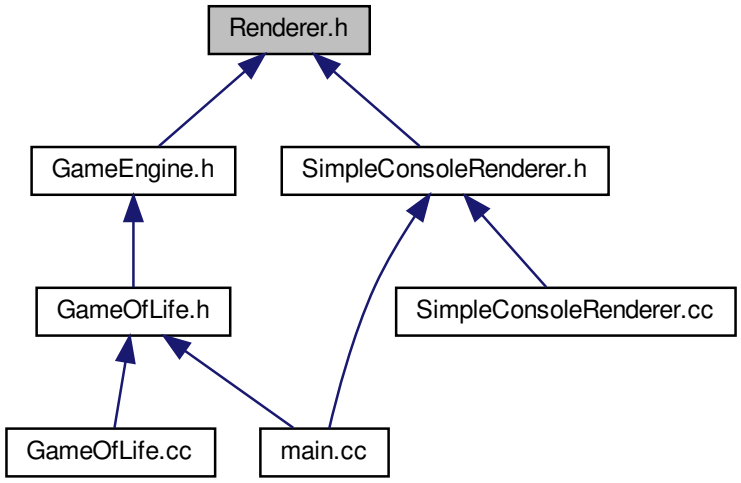
6.14 Renderer.h File Reference

```
#include <string>
```

Include dependency graph for Renderer.h:



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



Classes

- struct Coord

  Struct containing coordinates of different objects.
- class Renderer

  Basic base class for all renderers.

Enumerations

- enum class Color {
  Red , Green , Blue , Black ,
  White }

  Colors used in renderer.

6.14.1 Enumeration Type Documentation

6.14.1.1 Color

enum Color [strong]
Colors used in renderer.

Enumerator

Red	
Green	
Blue	
Black	
White	

Definition at line 20 of file Renderer.h.

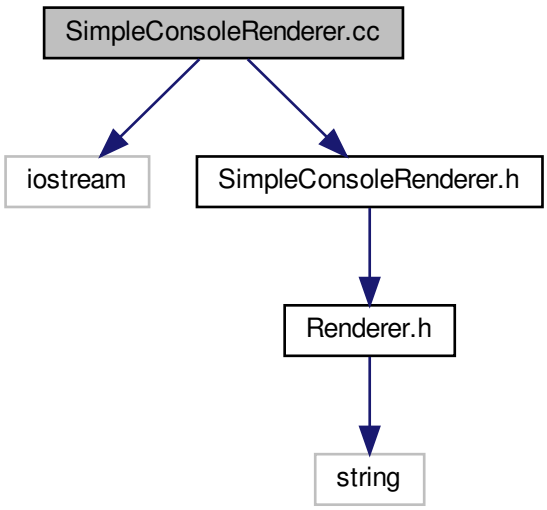
6.15 Renderer.h

```
00001 //
00002 // Created by mateu on 3/21/2021.
00003 //
00004
00005 #ifndef GAME_OF_LIFE_RENDERER_H
00006 #define GAME_OF_LIFE_RENDERER_H
00007
00008 #include <string>
00009
00011 struct Coord {
00012     Coord(int x_in, int y_in) :
00013             x(x_in),
00014             y(y_in) {}
00015     int x;
00016     int y;
00017 };
00018
00020 enum class Color {
00021     Red,
00022     Green,
00023     Blue,
00024     Black,
00025     White
00026 };
00027
00029 class Renderer {
00030 public:
00031
00035     virtual void create_window(int size_x, int size_y) = 0;
00036
00042     virtual void draw_square(const Coord &position, int size_x, int size_y, const Color &fill) = 0;
00043
00047     virtual void set_pixel(const Coord &position, const Color &fill) = 0;
00048
00051     virtual void clear_screen(const Color &fill) = 0;
00052
00056     virtual void show_text_big(const Coord &position, const std::string &text) = 0;
00057
00061     virtual void show_text_medium(const Coord &position, const std::string &text) = 0;
00062
00066     virtual void show_text_small(const Coord &position, const std::string &text) = 0;
00067
00068     virtual void render() = 0;
00069
00070     virtual ~Renderer() = default;
00071
00072 protected:
00074     int width_;
00076     int height_;
00077
00078 };
00079
00080
00081 #endif //GAME_OF_LIFE_RENDERER_H
```

6.16 SimpleConsoleRenderer.cc File Reference

```
#include <iostream>
#include "SimpleConsoleRenderer.h"
```

Include dependency graph for SimpleConsoleRenderer.cc:



Macros

- #define IS_ALPHA_NUMERIC(x) (x < 256)
- #define COLOR_BLACK 301
- #define COLOR_WHITE 300

6.16.1 Macro Definition Documentation

6.16.1.1 COLOR_BLACK

```
#define COLOR_BLACK 301
```

Definition at line 16 of file SimpleConsoleRenderer.cc.

6.16.1.2 COLOR_WHITE

```
#define COLOR_WHITE 300
```

Definition at line 17 of file SimpleConsoleRenderer.cc.

6.16.1.3 IS_ALPHA_NUMERIC

```
#define IS_ALPHA_NUMERIC(
            x ) (x < 256)
```

Definition at line 15 of file SimpleConsoleRenderer.cc.

6.17 SimpleConsoleRenderer.cc

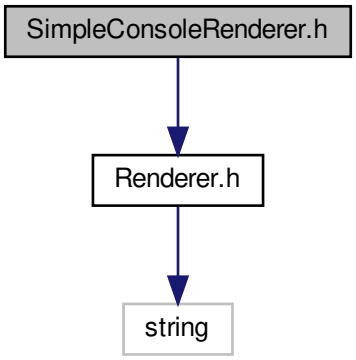
```
00001 //
00002 // Created by mateu on 4/11/2021.
00003 //
00004
00005 #include <iostream>
00006 #include "SimpleConsoleRenderer.h"
00007
00008
00009 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32) && !defined(__CYGWIN__)
00010
00011 #include <Windows.h>
00012
00013 #endif
00014
00015 #define IS_ALPHA_NUMERIC(x) (x < 256)
00016 #define COLOR_BLACK 301
00017 #define COLOR_WHITE 300
00018
00019
00020 void SimpleConsoleRenderer::create_window(int size_x, int size_y) {
00021     width_ = size_x;
00022     height_ = size_y;
00023     video_buffer_ = new GrayscalePixel[width_ * height_];
00024 }
00025
00026 void SimpleConsoleRenderer::draw_square(const Coord &position, int size_x, int size_y, const Color
       &fill) {
00027     throw std::exception("Not implemented");
00028 }
00029
00030 void SimpleConsoleRenderer::clear_screen(const Color &fill) {
00031
00032     memset(video_buffer_, color_to_pixel(fill), width_ * height_);
00033     clear_window();
00034 }
00035
00036 void SimpleConsoleRenderer::set_pixel(const Coord &position, const Color &fill) {
00037     video_buffer_[translate(position)] = color_to_pixel(fill);
00038 }
00039
00040 void SimpleConsoleRenderer::show_text_big(const Coord &position, const std::string &text) {
00041
00042     auto draw_stared_line = [this, &text, &position](int y) {
00043         for (int i = position.x; i < position.x + text.size() + 4; i++) {
00044             video_buffer_[translate({i, y})] = '*';
00045         }
00046     };
00047
00048     draw_stared_line(position.y);
00049
00050     int x = position.x;
00051     video_buffer_[translate({x++, position.y + 1})] = '*';
00052     video_buffer_[translate({x++, position.y + 1})] = ' ';
00053     for (int i = position.x + 1; i < position.x + text.size() + 1; i++) {
00054         video_buffer_[translate({i, position.y + 1})] = (unsigned char) text[i];
00055         x++;
00056     }
00057     video_buffer_[translate({x++, position.y + 1})] = ' ';
00058     video_buffer_[translate({x++, position.y + 1})] = '*';
00059
00060     draw_stared_line(position.y + 2);
00061 }
00062
00063 void SimpleConsoleRenderer::show_text_medium(const Coord &position, const std::string &text) {
00064     int x = position.x;
00065     video_buffer_[translate({x++, position.y})] = '*';
00066     video_buffer_[translate({x++, position.y})] = ' ';
00067     for (char i : text) {
00068         video_buffer_[translate({x++, position.y})] = (unsigned char) i;
00069     }
00070     video_buffer_[translate({x++, position.y})] = ' ';
00071     video_buffer_[translate({x++, position.y})] = '*';
00072 }
00073
00074 void SimpleConsoleRenderer::show_text_small(const Coord &position, const std::string &text) {
00075     for (int i = position.x; i < position.x + text.size(); i++) {
00076         video_buffer_[translate({i, position.y})] = (unsigned char) text[i];
00077     }
00078 }
00079
00080 void SimpleConsoleRenderer::render() {
00081     clear_window();
00082     for (int y = 0; y < height_; y++) {
00083       for (int x = 0; x < width_; x++) {
00084
```

```
00085         auto item = video_buffer_[translate({x, y})];
00086         if (IS_ALPHA_NUMERIC(item))
00087           std::cout « (char)item;
00088         else if (item >= COLOR_BLACK)
00089           std::cout « ' ';
00090         else if (item == COLOR_WHITE)
00091           std::cout « "#";
00092         else
00093           throw std::exception("Bad value");
00094
00095         std::cout « " ";
00096       }
00097       std::cout « std::endl;
00098     }
00099 }
00100
00101 unsigned SimpleConsoleRenderer::translate(Coord position) {
00102     return position.y * width_ + position.x;
00103 }
00104
00105 GrayscalePixel SimpleConsoleRenderer::color_to_pixel(const Color &color) {
00106     GrayscalePixel pixel_color = COLOR_BLACK;
00107     if (color == Color::White) {
00108         pixel_color = COLOR_WHITE;
00109     }
00110     return pixel_color;
00111 }
00112
00113 void SimpleConsoleRenderer::clear_window() {
00114 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32) && !defined(__CYGWIN__)
00115     system("cls");
00116 #else
00117     std::cout « "\x1B[2J\x1B[H";
00118 #endif
00119 }
```

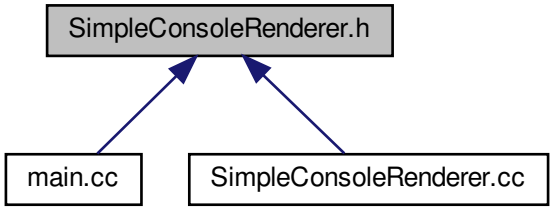
6.18 SimpleConsoleRenderer.h File Reference

`#include "Renderer.h"`

Include dependency graph for SimpleConsoleRenderer.h:



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



Classes

- class SimpleConsoleRenderer

Typedefs

- typedef uint16_t GrayscalePixel

6.18.1 Typedef Documentation

6.18.1.1 GrayscalePixel

```
typedef uint16_t GrayscalePixel
```

Definition at line 10 of file SimpleConsoleRenderer.h.

6.19 SimpleConsoleRenderer.h

```
00001 //
00002 // Created by mateu on 4/11/2021.
00003 //
00004
00005 #ifndef GAME_OF_LIFE_SIMPLECONSOLERENDERER_H
00006 #define GAME_OF_LIFE_SIMPLECONSOLERENDERER_H
00007
00008 #include "Renderer.h"
00009
00010 typedef uint16_t GrayscalePixel;
00011
00012 class SimpleConsoleRenderer : public Renderer {
00013
00014     unsigned translate(Coord position);
00015     static GrayscalePixel color_to_pixel(const Color& color);
00016
00017     void create_window(int size_x, int size_y) override;
00018
00019     void draw_square(const Coord &position, int size_x, int size_y, const Color &fill) override;
00020
00021     void clear_screen(const Color &fill) override;
00022
00023     void set_pixel(const Coord &position, const Color &fill) override;
00024
00025     void show_text_big(const Coord &position, const std::string &text) override;
00026
00027     void show_text_medium(const Coord &position, const std::string &text) override;
00028
00029     void show_text_small(const Coord &position, const std::string &text) override;
00030
00031     void render() override;
00032
00033     ~SimpleConsoleRenderer() override = default;
```

```
00034
00035 private:
00036
00037     void clear_window();
00038
00039     GrayscalePixel* video_buffer_;
00040
00041 };
00042
00043 #endif // GAME_OF_LIFE_SIMPLECONSOLERENDERER_H
```

Index