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

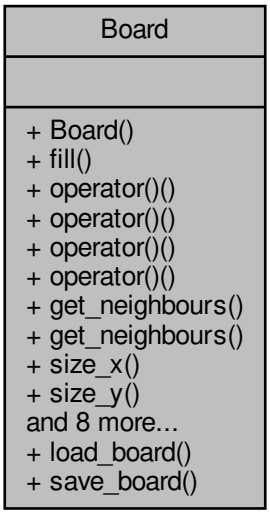
Here is a list of all files with brief descriptions:

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

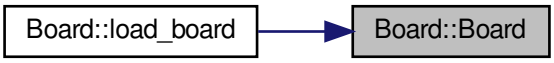
Class containing board (bool array) with functions useful for the implementation of the game of life
Definition at line 14 of file Board.h.

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()

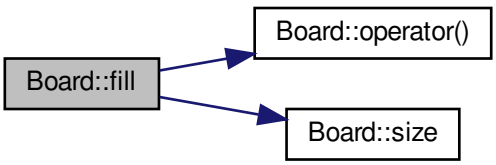
```
void Board::fill (
            bool value )
```

Fils the board wirh specified value

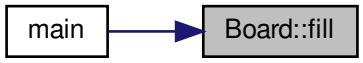
Parameters

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()

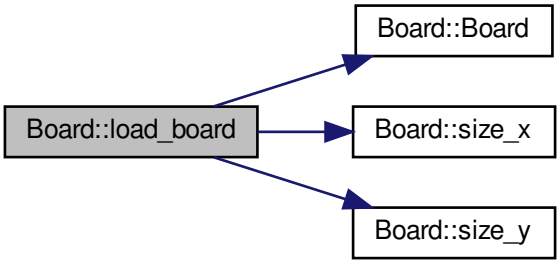
```
Board Board::load_board (
            const std::string & path )  [static]
```

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

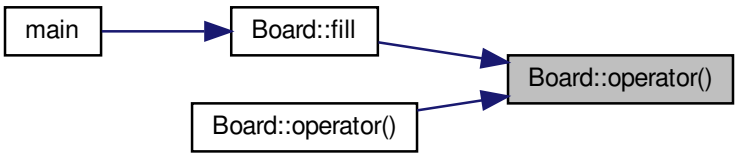
Parameters

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()

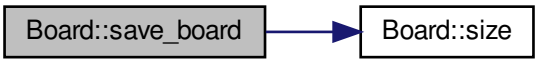
```
void Board::save_board (
            const Board & board,
            const std::string & path )  [static]
```

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:

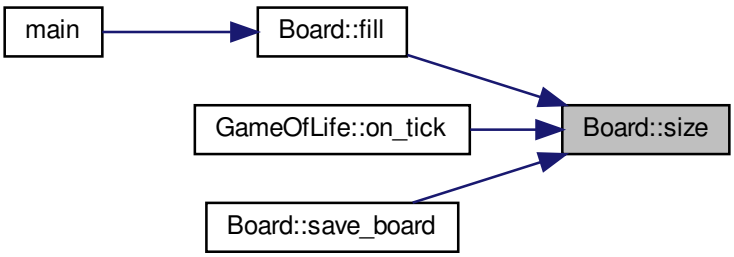


5.1.3.12 size()

```
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()

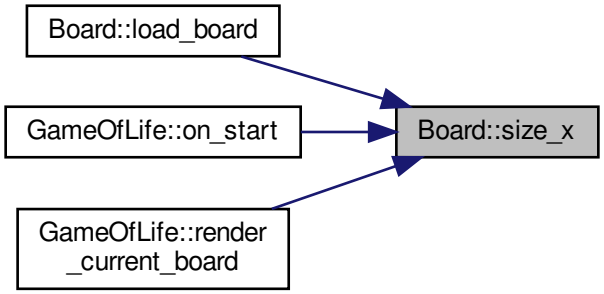
`size_t Board::size_x ( ) const`

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()

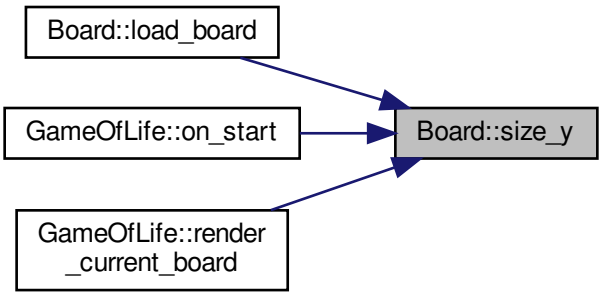
`size_t Board::size_y ( ) const`

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:



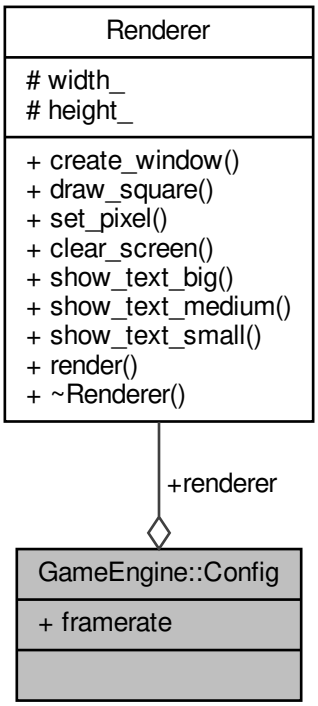
The documentation for this class was generated from the following files:

- 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

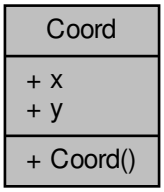
```
Renderer* GameEngine::Config::renderer
```

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.

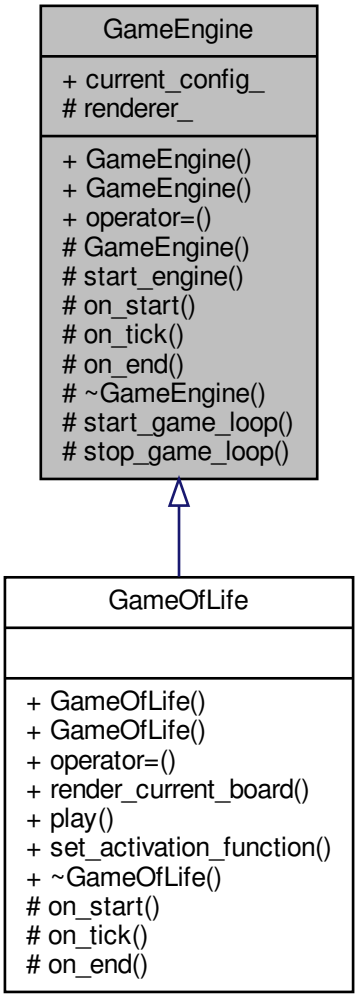
5.3.3.2 y

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

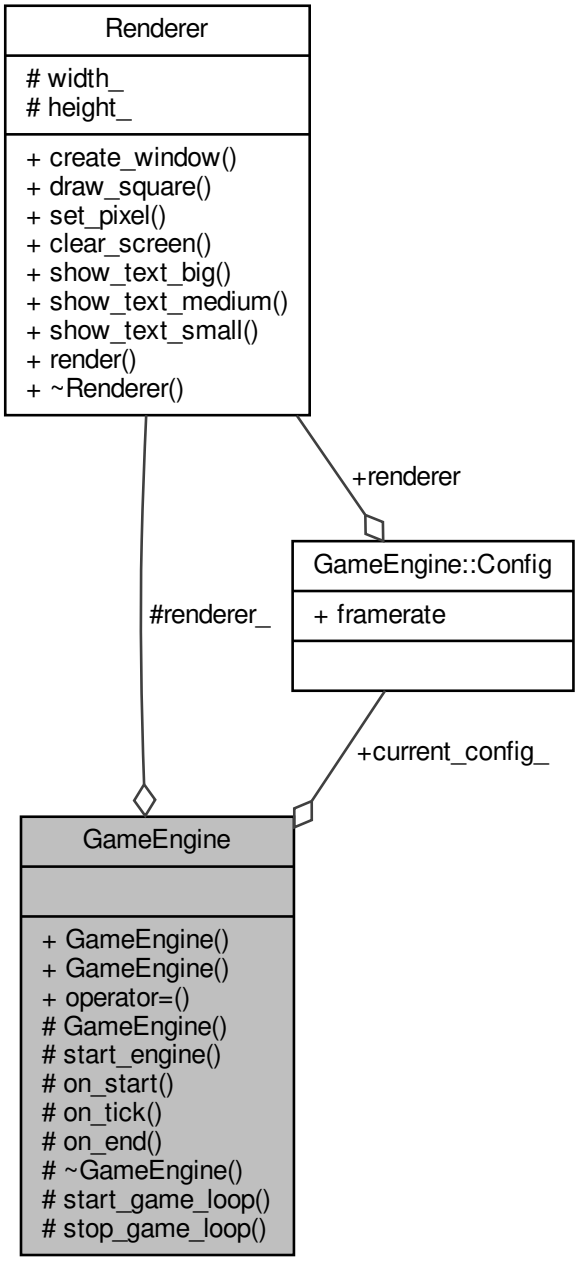
- Renderer.h

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]
```

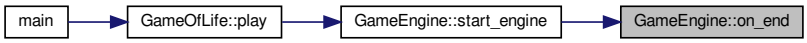
Definition at line 66 of file GameEngine.h.

5.4.3 Member Function Documentation

5.4.3.1 on_end()

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

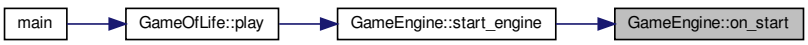
This function is called when the game ends should be ovverided by the derriving class.
Implemented in GameOfLife.
Here is the caller graph for this function:



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:

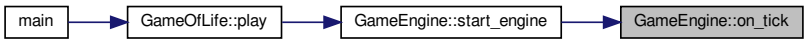


5.4.3.3 on_tick()

```
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:



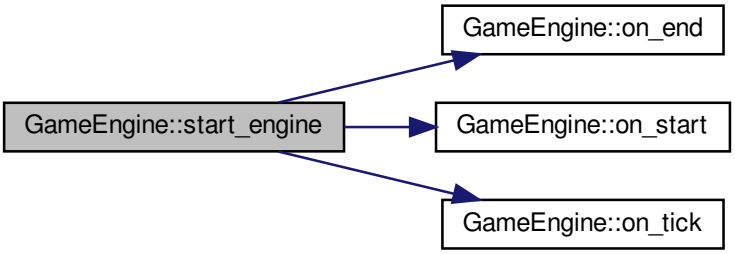
5.4.3.4 operator=()

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

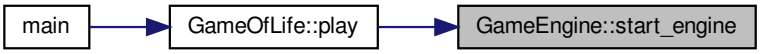
5.4.3.5 start_engine()

```
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:

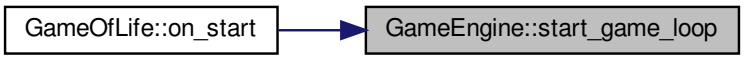


5.4.3.6 start_game_loop()

```
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]
```

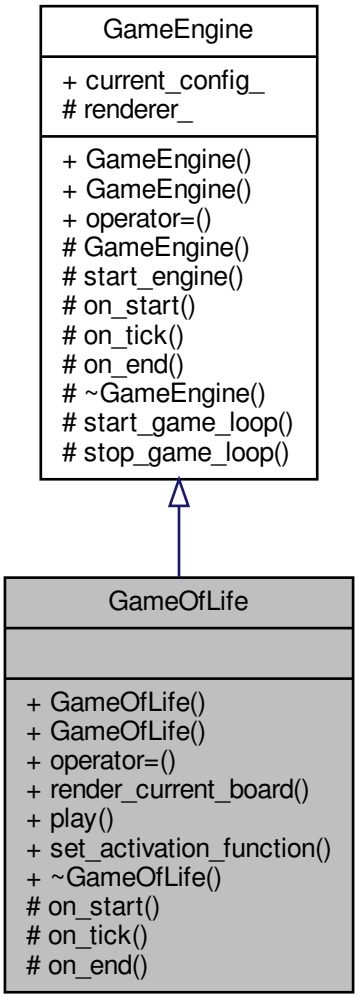
Definition at line 75 of file GameEngine.h.
The documentation for this class was generated from the following file:

- GameEngine.h

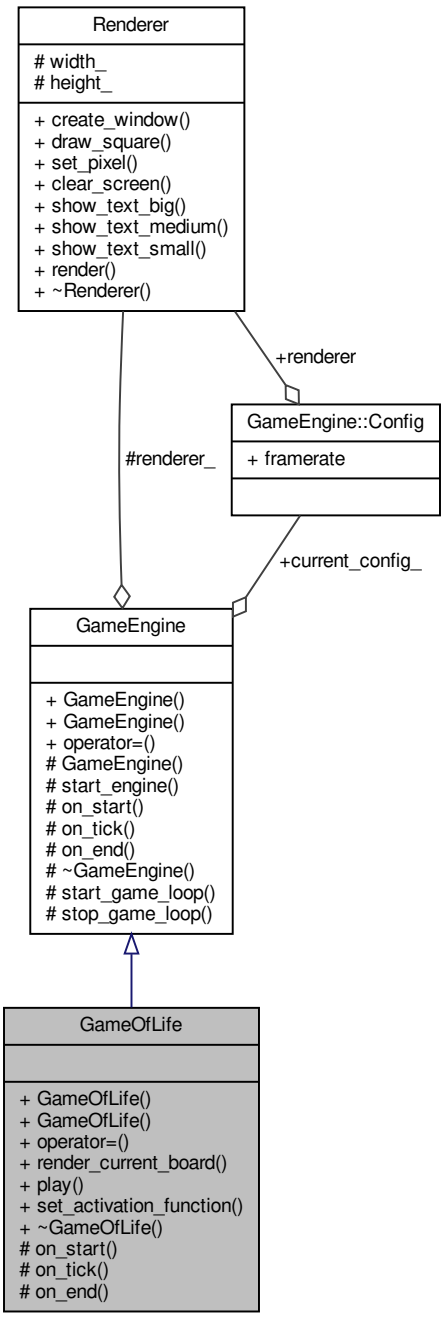
5.5 GameOfLife Class Reference

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.

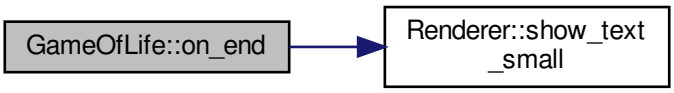
5.5.3 Member Function Documentation

5.5.3.1 on_end()

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

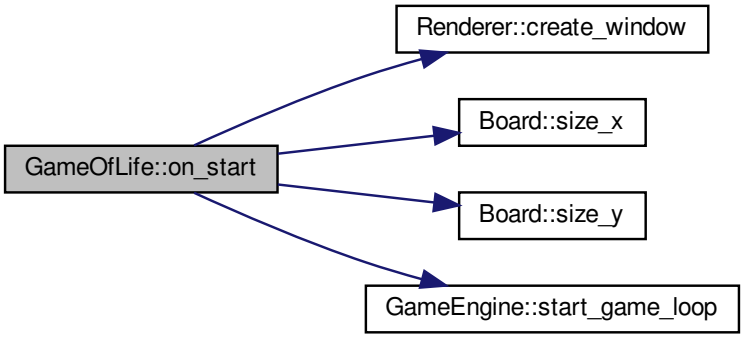
This function is called when the game ends should be ovverided by the derriving class.
Implements GameEngine.

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



5.5.3.2 on_start()

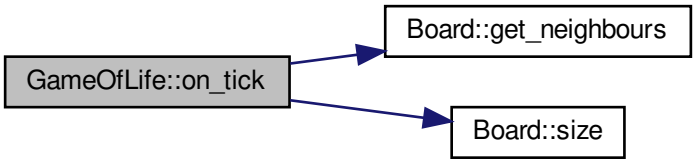
`void GameOfLife::on_start ( ) [override], [protected], [virtual]`
This function is called on game start should be overrided by the deriving class.
Implements GameEngine.
Definition at line 33 of file GameOfLife.cc.
Here is the call graph for this function:



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:



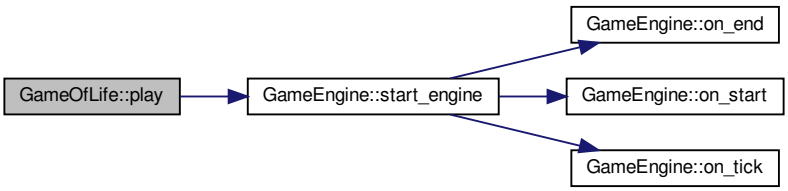
5.5.3.4 operator=()

```
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:

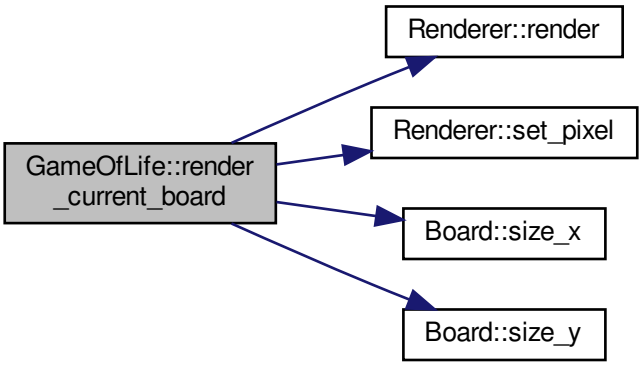


Here is the caller graph for this function:

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

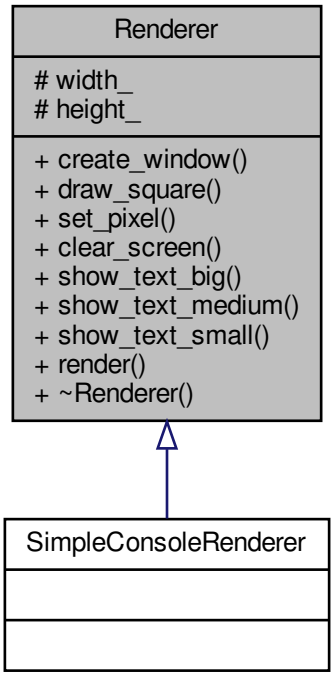
Definition at line 22 of file GameOfLife.cc.
The documentation for this class was generated from the following files:

- GameOfLife.h
- GameOfLife.cc

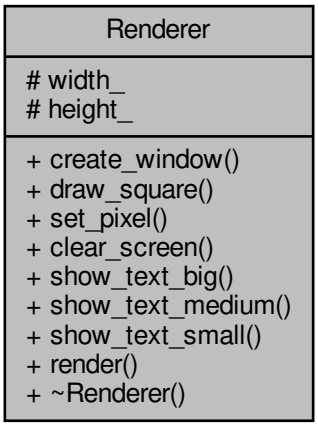
5.6 Renderer Class Reference

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.

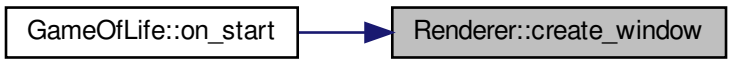
Parameters

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.

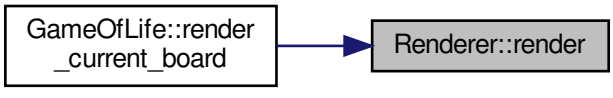
Parameters

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 (
```

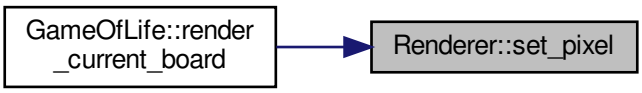
```
            const Coord & position,
            const Color & fill )  [pure virtual]
```

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()

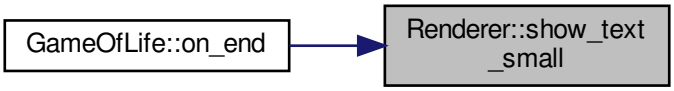
```
virtual void Renderer::show_text_small (
            const Coord & position,
            const std::string & text )  [pure virtual]
```

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_

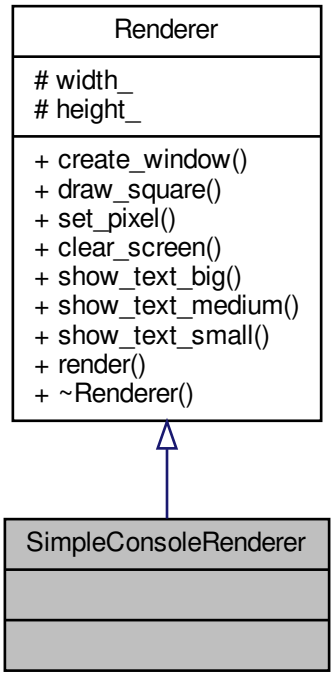
`int Renderer::width_ [protected]`
width of the render plane
Definition at line 74 of file Renderer.h.
The documentation for this class was generated from the following file:

- Renderer.h

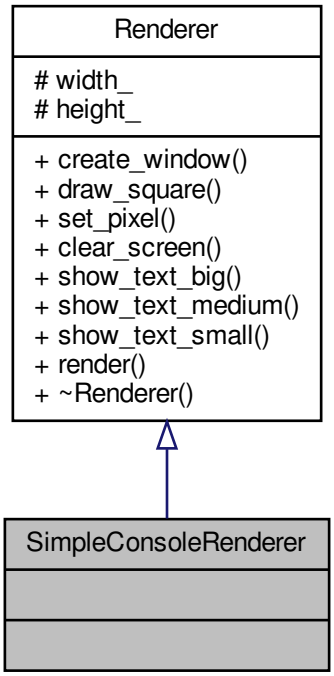
5.7 SimpleConsoleRenderer Class Reference

`#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

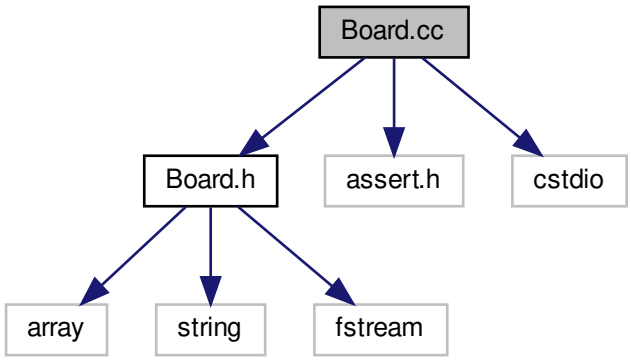
Chapter 6

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;
```

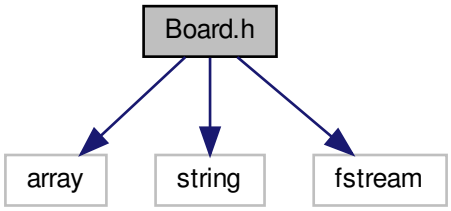
```
00022 }
00023
00024 size_t Board::size_x() const {
00025     return size_x_;
00026 }
00027
00028 size_t Board::size_y() const {
00029     return size_y_;
00030 }
00031
00032 void Board::fill(bool value) {
00033   for (int i = 0; i < size(); i++) {
00034     operator()(i) = value;
00035   }
00036 }
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 }
```

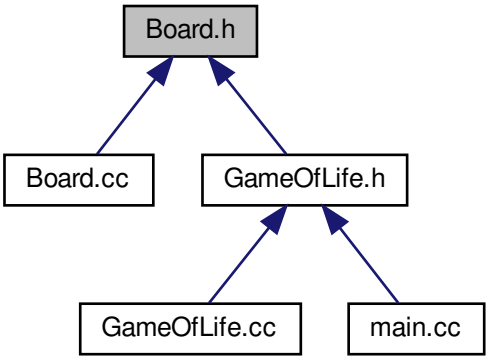
6.3 Board.h File Reference

```
#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 build/CMakeFiles/3.19.7/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_DIALECT

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"

- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_dialect_default

6.5.1 Macro Definition Documentation

6.5.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 561 of file CMakeCCompilerId.c.

6.5.1.2 C_DIALECT

```
#define C_DIALECT
```

Definition at line 645 of file CMakeCCompilerId.c.

6.5.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 314 of file CMakeCCompilerId.c.

6.5.1.4 DEC

```
#define DEC(
              n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 565 of file CMakeCCompilerId.c.

6.5.1.5 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 576 of file CMakeCCompilerId.c.

6.5.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 439 of file CMakeCCompilerId.c.

6.5.1.7 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 335 of file CMakeCCompilerId.c.

6.5.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 334 of file CMakeCCompilerId.c.

6.5.2 Function Documentation

6.5.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 665 of file CMakeCCompilerId.c.

6.5.3 Variable Documentation

6.5.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 636 of file CMakeCCompilerId.c.

6.5.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 321 of file CMakeCCompilerId.c.

6.5.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
=
  "INFO" ":" "dialect_default[" C_DIALECT "]"
```

Definition at line 654 of file CMakeCCompilerId.c.

6.5.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 635 of file CMakeCCompilerId.c.

6.6 CMakeCCompilerId.c

```
00001 #ifdef __cplusplus
00002 # error "A C++ compiler has been selected for C."
00003 #endif
00004
00005 #if defined(__18CXX)
```

```
00006 # define ID_VOID_MAIN
00007 #endif
00008 #if defined(__CLASSIC_C__)
00009 /* cv-qualifiers did not exist in K&R C */
00010 # define const
00011 # define volatile
00012 #endif
00013
00014
00015 /* Version number components: V=Version, R=Revision, P=Patch
00016    Version date components:   YYYY=Year, MM=Month,   DD=Day  */
00017
00018 #if defined(__INTEL_COMPILER) || defined(__ICC)
00019 # define COMPILER_ID "Intel"
00020 # if defined(_MSC_VER)
00021 #  define SIMULATE_ID "MSVC"
00022 # endif
00023 # if defined(__GNUC__)
00024 #  define SIMULATE_ID "GNU"
00025 # endif
00026   /* __INTEL_COMPILER = VRP */
00027 # define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00028 # define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00029 # if defined(__INTEL_COMPILER_UPDATE)
00030 #  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00031 # else
00032 #  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
00033 # endif
00034 # if defined(__INTEL_COMPILER_BUILD_DATE)
00035   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00036 #  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00037 # endif
00038 # if defined(_MSC_VER)
00039    /* _MSC_VER = VVRR */
00040 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00041 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00042 # endif
00043 # if defined(__GNUC__)
00044 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00045 # elif defined(__GNUG__)
00046 #  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00047 # endif
00048 # if defined(__GNUC_MINOR__)
00049 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00050 # endif
00051 # if defined(__GNUC_PATCHLEVEL__)
00052 #  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00053 # endif
00054
00055 #elif defined(__PATHCC__)
00056 # define COMPILER_ID "PathScale"
00057 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00058 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00059 # if defined(__PATHCC_PATCHLEVEL__)
00060 #  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00061 # endif
00062
00063 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
00064 # define COMPILER_ID "Embarcadero"
00065 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
00066 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
00067 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)
00068
00069 #elif defined(__BORLANDC__)
00070 # define COMPILER_ID "Borland"
00071   /* __BORLANDC__ = 0xVRR */
00072 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
00073 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00074
00075 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00076 # define COMPILER_ID "Watcom"
00077    /* __WATCOMC__ = VVRR */
00078 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00079 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00080 # if (__WATCOMC__ % 10) > 0
00081 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00082 # endif
00083
00084 #elif defined(__WATCOMC__)
00085 # define COMPILER_ID "OpenWatcom"
00086    /* __WATCOMC__ = VVRP + 1100 */
00087 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00088 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00089 # if (__WATCOMC__ % 10) > 0
00090 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00091 # endif
00092
```

```
00093 #elif defined(__SUNPRO_C)
00094 # define COMPILER_ID "SunPro"
00095 # if __SUNPRO_C >= 0x5100
00096   /* __SUNPRO_C = 0xVRRP */
00097 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»12)
00098 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xFF)
00099 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00100 # else
00101   /* __SUNPRO_CC = 0xVRP */
00102 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_C»8)
00103 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_C»4 & 0xF)
00104 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_C    & 0xF)
00105 # endif
00106
00107 #elif defined(__HP_cc)
00108 # define COMPILER_ID "HP"
00109   /* __HP_cc = VVRRPP */
00110 # define COMPILER_VERSION_MAJOR DEC(__HP_cc/10000)
00111 # define COMPILER_VERSION_MINOR DEC(__HP_cc/100 % 100)
00112 # define COMPILER_VERSION_PATCH DEC(__HP_cc     % 100)
00113
00114 #elif defined(__DECC)
00115 # define COMPILER_ID "Compaq"
00116   /* __DECC_VER = VVRRTPPPP */
00117 # define COMPILER_VERSION_MAJOR DEC(__DECC_VER/10000000)
00118 # define COMPILER_VERSION_MINOR DEC(__DECC_VER/100000  % 100)
00119 # define COMPILER_VERSION_PATCH DEC(__DECC_VER         % 10000)
00120
00121 #elif defined(__IBMC__) && defined(__COMPILER_VER__)
00122 # define COMPILER_ID "zOS"
00123   /* __IBMC__ = VRP */
00124 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00125 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00126 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00127
00128 #elif defined(__ibmxl__) && defined(__clang__)
00129 # define COMPILER_ID "XLClang"
00130 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00131 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00132 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00133 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00134
00135
00136 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ >= 800
00137 # define COMPILER_ID "XL"
00138   /* __IBMC__ = VRP */
00139 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00140 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00141 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00142
00143 #elif defined(__IBMC__) && !defined(__COMPILER_VER__) && __IBMC__ < 800
00144 # define COMPILER_ID "VisualAge"
00145   /* __IBMC__ = VRP */
00146 # define COMPILER_VERSION_MAJOR DEC(__IBMC__/100)
00147 # define COMPILER_VERSION_MINOR DEC(__IBMC__/10 % 10)
00148 # define COMPILER_VERSION_PATCH DEC(__IBMC__    % 10)
00149
00150 #elif defined(__PGI)
00151 # define COMPILER_ID "PGI"
00152 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00153 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00154 # if defined(__PGIC_PATCHLEVEL__)
00155 #  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00156 # endif
00157
00158 #elif defined(_CRAYC)
00159 # define COMPILER_ID "Cray"
00160 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00161 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00162
00163 #elif defined(__TI_COMPILER_VERSION__)
00164 # define COMPILER_ID "TI"
00165   /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
00166 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00167 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
00168 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)
00169
00170 #elif defined(__FUJITSU) || defined(__FCC_VERSION) || defined(__fcc_version)
00171 # define COMPILER_ID "Fujitsu"
00172
00173 #elif defined(__ghs__)
00174 # define COMPILER_ID "GHS"
00175 /* __GHS_VERSION_NUMBER = VVVVRP */
00176 # ifdef __GHS_VERSION_NUMBER
00177 # define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00178 # define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
```

```
00180 # endif
00181
00182 #elif defined(__TINYC__)
00183 # define COMPILER_ID "TinyCC"
00184
00185 #elif defined(__BCC__)
00186 # define COMPILER_ID "Bruce"
00187
00188 #elif defined(__SCO_VERSION__)
00189 # define COMPILER_ID "SCO"
00190
00191 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00192 # define COMPILER_ID "ARMCC"
00193 #if __ARMCC_VERSION >= 1000000
00194   /* __ARMCC_VERSION = VRRPPPP */
00195   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00196   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00197   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
00198 #else
00199   /* __ARMCC_VERSION = VRPPPP */
00200   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00201   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00202   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
00203 #endif
00204
00205
00206 #elif defined(__clang__) && defined(__apple_build_version__)
00207 # define COMPILER_ID "AppleClang"
00208 # if defined(_MSC_VER)
00209 #  define SIMULATE_ID "MSVC"
00210 # endif
00211 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00212 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00213 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00214 # if defined(_MSC_VER)
00215   /* _MSC_VER = VVRR */
00216 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00217 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00218 # endif
00219 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00220
00221 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00222 # define COMPILER_ID "ARMClang"
00223   # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00224   # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00225   # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
00226 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00227
00228 #elif defined(__clang__)
00229 # define COMPILER_ID "Clang"
00230 # if defined(_MSC_VER)
00231 #  define SIMULATE_ID "MSVC"
00232 # endif
00233 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00234 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00235 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00236 # if defined(_MSC_VER)
00237   /* _MSC_VER = VVRR */
00238 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00239 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00240 # endif
00241
00242 #elif defined(__GNUC__)
00243 # define COMPILER_ID "GNU"
00244 # define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00245 # if defined(__GNUC_MINOR__)
00246 #  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00247 # endif
00248 # if defined(__GNUC_PATCHLEVEL__)
00249 #  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00250 # endif
00251
00252 #elif defined(_MSC_VER)
00253 # define COMPILER_ID "MSVC"
00254   /* _MSC_VER = VVRR */
00255 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00256 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00257 # if defined(_MSC_FULL_VER)
00258 #  if _MSC_VER >= 1400
00259     /* _MSC_FULL_VER = VVRRPPPPP */
00260 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00261 #  else
00262     /* _MSC_FULL_VER = VVRRPPPP */
00263 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00264 #  endif
00265 # endif
00266 # if defined(_MSC_BUILD)
```

```
00267 #  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00268 # endif
00269
00270 #elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__ADSPTS__) ||
       defined(__ADSP21000__)
00271 # define COMPILER_ID "ADSP"
00272 #if defined(__VISUALDSPVERSION__)
00273   /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
00274 # define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__»24)
00275 # define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__»16 & 0xFF)
00276 # define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__»8  & 0xFF)
00277 #endif
00278
00279 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00280 # define COMPILER_ID "IAR"
00281 # if defined(__VER__) && defined(__ICCARM__)
00282 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00283 #  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00284 #  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00285 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00286 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
       defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
       defined(__ICC8051__))
00287 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00288 #  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00289 #  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00290 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00291 # endif
00292
00293 #elif defined(__SDCC_VERSION_MAJOR) || defined(SDCC)
00294 # define COMPILER_ID "SDCC"
00295 # if defined(__SDCC_VERSION_MAJOR)
00296 #  define COMPILER_VERSION_MAJOR DEC(__SDCC_VERSION_MAJOR)
00297 #  define COMPILER_VERSION_MINOR DEC(__SDCC_VERSION_MINOR)
00298 #  define COMPILER_VERSION_PATCH DEC(__SDCC_VERSION_PATCH)
00299 # else
00300   /* SDCC = VRP */
00301 #  define COMPILER_VERSION_MAJOR DEC(SDCC/100)
00302 #  define COMPILER_VERSION_MINOR DEC(SDCC/10 % 10)
00303 #  define COMPILER_VERSION_PATCH DEC(SDCC    % 10)
00304 # endif
00305
00306
00307 /* These compilers are either not known or too old to define an
00308   identification macro.  Try to identify the platform and guess that
00309   it is the native compiler.  */
00310 #elif defined(__hpux) || defined(__hpua)
00311 # define COMPILER_ID "HP"
00312
00313 #else /* unknown compiler */
00314 # define COMPILER_ID ""
00315 #endif
00316
00317 /* Construct the string literal in pieces to prevent the source from
00318    getting matched.  Store it in a pointer rather than an array
00319    because some compilers will just produce instructions to fill the
00320    array rather than assigning a pointer to a static array.  */
00321 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
00322 #ifdef SIMULATE_ID
00323 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
00324 #endif
00325
00326 #ifdef __QNXNTO__
00327 char const* qnxnto = "INFO" ":" "qnxnto[]";
00328 #endif
00329
00330 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00331 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00332 #endif
00333
00334 #define STRINGIFY_HELPER(X) #X
00335 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00336
00337 /* Identify known platforms by name.  */
00338 #if defined(__linux) || defined(__linux__) || defined(linux)
00339 # define PLATFORM_ID "Linux"
00340
00341 #elif defined(__CYGWIN__)
00342 # define PLATFORM_ID "Cygwin"
00343
00344 #elif defined(__MINGW32__)
00345 # define PLATFORM_ID "MinGW"
00346
00347 #elif defined(__APPLE__)
00348 # define PLATFORM_ID "Darwin"
00349
00350 #elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
```

```
00351 # define PLATFORM_ID "Windows"
00352
00353 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00354 # define PLATFORM_ID "FreeBSD"
00355
00356 #elif defined(__NetBSD__) || defined(__NetBSD)
00357 # define PLATFORM_ID "NetBSD"
00358
00359 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00360 # define PLATFORM_ID "OpenBSD"
00361
00362 #elif defined(__sun) || defined(sun)
00363 # define PLATFORM_ID "SunOS"
00364
00365 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00366 # define PLATFORM_ID "AIX"
00367
00368 #elif defined(__hpux) || defined(__hpux__)
00369 # define PLATFORM_ID "HP-UX"
00370
00371 #elif defined(__HAIKU__)
00372 # define PLATFORM_ID "Haiku"
00373
00374 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00375 # define PLATFORM_ID "BeOS"
00376
00377 #elif defined(__QNX__) || defined(__QNXNTO__)
00378 # define PLATFORM_ID "QNX"
00379
00380 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00381 # define PLATFORM_ID "Tru64"
00382
00383 #elif defined(__riscos) || defined(__riscos__)
00384 # define PLATFORM_ID "RISCos"
00385
00386 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00387 # define PLATFORM_ID "SINIX"
00388
00389 #elif defined(__UNIX_SV__)
00390 # define PLATFORM_ID "UNIX_SV"
00391
00392 #elif defined(__bsdos__)
00393 # define PLATFORM_ID "BSDOS"
00394
00395 #elif defined(_MPRAS) || defined(MPRAS)
00396 # define PLATFORM_ID "MP-RAS"
00397
00398 #elif defined(__osf) || defined(__osf__)
00399 # define PLATFORM_ID "OSF1"
00400
00401 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00402 # define PLATFORM_ID "SCO_SV"
00403
00404 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00405 # define PLATFORM_ID "ULTRIX"
00406
00407 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00408 # define PLATFORM_ID "Xenix"
00409
00410 #elif defined(__WATCOMC__)
00411 # if defined(__LINUX__)
00412 #  define PLATFORM_ID "Linux"
00413
00414 # elif defined(__DOS__)
00415 #  define PLATFORM_ID "DOS"
00416
00417 # elif defined(__OS2__)
00418 #  define PLATFORM_ID "OS2"
00419
00420 # elif defined(__WINDOWS__)
00421 #  define PLATFORM_ID "Windows3x"
00422
00423 # elif defined(__VXWORKS__)
00424 #  define PLATFORM_ID "VxWorks"
00425
00426 # else /* unknown platform */
00427 #  define PLATFORM_ID
00428 # endif
00429
00430 #elif defined(__INTEGRITY)
00431 # if defined(INT_178B)
00432 #  define PLATFORM_ID "Integrity178"
00433
00434 # else /* regular Integrity */
00435 #  define PLATFORM_ID "Integrity"
00436 # endif
00437
```

```
00438 #else /* unknown platform */
00439 # define PLATFORM_ID
00440
00441 #endif
00442
00443 /* For windows compilers MSVC and Intel we can determine
00444    the architecture of the compiler being used.  This is because
00445    the compilers do not have flags that can change the architecture,
00446    but rather depend on which compiler is being used
00447 */
00448 #if defined(_WIN32) && defined(_MSC_VER)
00449 # if defined(_M_IA64)
00450 #  define ARCHITECTURE_ID "IA64"
00451
00452 # elif defined(_M_X64) || defined(_M_AMD64)
00453 #  define ARCHITECTURE_ID "x64"
00454
00455 # elif defined(_M_IX86)
00456 #  define ARCHITECTURE_ID "X86"
00457
00458 # elif defined(_M_ARM64)
00459 #  define ARCHITECTURE_ID "ARM64"
00460
00461 # elif defined(_M_ARM)
00462 #  if _M_ARM == 4
00463 #   define ARCHITECTURE_ID "ARMV4I"
00464 #  elif _M_ARM == 5
00465 #   define ARCHITECTURE_ID "ARMV5I"
00466 #  else
00467 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00468 #  endif
00469
00470 # elif defined(_M_MIPS)
00471 #  define ARCHITECTURE_ID "MIPS"
00472
00473 # elif defined(_M_SH)
00474 #  define ARCHITECTURE_ID "SHx"
00475
00476 # else /* unknown architecture */
00477 #  define ARCHITECTURE_ID ""
00478 # endif
00479
00480 #elif defined(__WATCOMC__)
00481 # if defined(_M_I86)
00482 #  define ARCHITECTURE_ID "I86"
00483
00484 # elif defined(_M_IX86)
00485 #  define ARCHITECTURE_ID "X86"
00486
00487 # else /* unknown architecture */
00488 #  define ARCHITECTURE_ID ""
00489 # endif
00490
00491 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00492 # if defined(__ICCARM__)
00493 #  define ARCHITECTURE_ID "ARM"
00494
00495 # elif defined(__ICCRX__)
00496 #  define ARCHITECTURE_ID "RX"
00497
00498 # elif defined(__ICCRH850__)
00499 #  define ARCHITECTURE_ID "RH850"
00500
00501 # elif defined(__ICCRL78__)
00502 #  define ARCHITECTURE_ID "RL78"
00503
00504 # elif defined(__ICCRISCV__)
00505 #  define ARCHITECTURE_ID "RISCV"
00506
00507 # elif defined(__ICCAVR__)
00508 #  define ARCHITECTURE_ID "AVR"
00509
00510 # elif defined(__ICC430__)
00511 #  define ARCHITECTURE_ID "MSP430"
00512
00513 # elif defined(__ICCV850__)
00514 #  define ARCHITECTURE_ID "V850"
00515
00516 # elif defined(__ICC8051__)
00517 #  define ARCHITECTURE_ID "8051"
00518
00519 # else /* unknown architecture */
00520 #  define ARCHITECTURE_ID ""
00521 # endif
00522
00523 #elif defined(__ghs__)
00524 # if defined(__PPC64__)
```

```
00525 #  define ARCHITECTURE_ID "PPC64"
00526
00527 # elif defined(__ppc__)
00528 #  define ARCHITECTURE_ID "PPC"
00529
00530 # elif defined(__ARM__)
00531 #  define ARCHITECTURE_ID "ARM"
00532
00533 # elif defined(__x86_64__)
00534 #  define ARCHITECTURE_ID "x64"
00535
00536 # elif defined(__i386__)
00537 #  define ARCHITECTURE_ID "X86"
00538
00539 # else /* unknown architecture */
00540 #  define ARCHITECTURE_ID ""
00541 # endif
00542
00543 #elif defined(__TI_COMPILER_VERSION__)
00544 # if defined(__TI_ARM__)
00545 #  define ARCHITECTURE_ID "ARM"
00546
00547 # elif defined(__MSP430__)
00548 #  define ARCHITECTURE_ID "MSP430"
00549
00550 # elif defined(__TMS320C28XX__)
00551 #  define ARCHITECTURE_ID "TMS320C28x"
00552
00553 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00554 #  define ARCHITECTURE_ID "TMS320C6x"
00555
00556 # else /* unknown architecture */
00557 #  define ARCHITECTURE_ID ""
00558 # endif
00559
00560 #else
00561 #  define ARCHITECTURE_ID
00562 #endif
00563
00564 /* Convert integer to decimal digit literals.  */
00565 #define DEC(n)                   \
00566   ('0' + (((n) / 10000000)%10)), \
00567   ('0' + (((n) / 1000000)%10)),  \
00568   ('0' + (((n) / 100000)%10)),   \
00569   ('0' + (((n) / 10000)%10)),    \
00570   ('0' + (((n) / 1000)%10)),     \
00571   ('0' + (((n) / 100)%10)),      \
00572   ('0' + (((n) / 10)%10)),       \
00573   ('0' +  ((n) % 10))
00574
00575 /* Convert integer to hex digit literals.  */
00576 #define HEX(n)             \
00577   ('0' + ((n)>>28 & 0xF)), \
00578   ('0' + ((n)>>24 & 0xF)), \
00579   ('0' + ((n)>>20 & 0xF)), \
00580   ('0' + ((n)>>16 & 0xF)), \
00581   ('0' + ((n)>>12 & 0xF)), \
00582   ('0' + ((n)>>8  & 0xF)), \
00583   ('0' + ((n)>>4  & 0xF)), \
00584   ('0' + ((n)     & 0xF))
00585
00586 /* Construct a string literal encoding the version number components. */
00587 #ifdef COMPILER_VERSION_MAJOR
00588 char const info_version[] = {
00589   'I', 'N', 'F', 'O', ':',
00590   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
00591   COMPILER_VERSION_MAJOR,
00592 # ifdef COMPILER_VERSION_MINOR
00593   '.', COMPILER_VERSION_MINOR,
00594 #  ifdef COMPILER_VERSION_PATCH
00595    '.', COMPILER_VERSION_PATCH,
00596 #   ifdef COMPILER_VERSION_TWEAK
00597     '.', COMPILER_VERSION_TWEAK,
00598 #   endif
00599 #  endif
00600 # endif
00601   ']','\0'};
00602 #endif
00603
00604 /* Construct a string literal encoding the internal version number. */
00605 #ifdef COMPILER_VERSION_INTERNAL
00606 char const info_version_internal[] = {
00607   'I', 'N', 'F', 'O', ':',
00608   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','_',
00609   'i','n','t','e','r','n','a','l','[',
00610   COMPILER_VERSION_INTERNAL,']','\0'};
00611 #endif
```

```
00612
00613 /* Construct a string literal encoding the version number components. */
00614 #ifdef SIMULATE_VERSION_MAJOR
00615 char const info_simulate_version[] = {
00616   'I', 'N', 'F', 'O', ':',
00617   's','i','m','u','l','a','t','e','_','v','e','r','s','i','o','n','[',
00618   SIMULATE_VERSION_MAJOR,
00619 # ifdef SIMULATE_VERSION_MINOR
00620   '.', SIMULATE_VERSION_MINOR,
00621 #  ifdef SIMULATE_VERSION_PATCH
00622    '.', SIMULATE_VERSION_PATCH,
00623 #   ifdef SIMULATE_VERSION_TWEAK
00624     '.', SIMULATE_VERSION_TWEAK,
00625 #   endif
00626 #  endif
00627 # endif
00628   ']','\0'};
00629 #endif
00630
00631 /* Construct the string literal in pieces to prevent the source from
00632    getting matched.  Store it in a pointer rather than an array
00633    because some compilers will just produce instructions to fill the
00634    array rather than assigning a pointer to a static array.  */
00635 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
00636 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
00637
00638
00639
00640 #if !defined(__STDC__)
00641 # if (defined(_MSC_VER) && !defined(__clang__)) \
00642   || (defined(__ibmxl__) || defined(__IBMC__))
00643 #  define C_DIALECT "90"
00644 # else
00645 #  define C_DIALECT
00646 # endif
00647 #elif __STDC_VERSION__ >= 201000L
00648 # define C_DIALECT "11"
00649 #elif __STDC_VERSION__ >= 199901L
00650 # define C_DIALECT "99"
00651 #else
00652 # define C_DIALECT "90"
00653 #endif
00654 const char* info_language_dialect_default =
00655   "INFO" ":" "dialect_default[" C_DIALECT "]";
00656
00657 /*--------------------------------------------------------------------------*/
00658
00659 #ifdef ID_VOID_MAIN
00660 void main() {}
00661 #else
00662 # if defined(__CLASSIC_C__)
00663 int main(argc, argv) int argc; char *argv[];
00664 # else
00665 int main(int argc, char* argv[])
00666 # endif
00667 {
00668   int require = 0;
00669   require += info_compiler[argc];
00670   require += info_platform[argc];
00671   require += info_arch[argc];
00672 #ifdef COMPILER_VERSION_MAJOR
00673   require += info_version[argc];
00674 #endif
00675 #ifdef COMPILER_VERSION_INTERNAL
00676   require += info_version_internal[argc];
00677 #endif
00678 #ifdef SIMULATE_ID
00679   require += info_simulate[argc];
00680 #endif
00681 #ifdef SIMULATE_VERSION_MAJOR
00682   require += info_simulate_version[argc];
00683 #endif
00684 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00685   require += info_cray[argc];
00686 #endif
00687   require += info_language_dialect_default[argc];
00688   (void)argv;
00689   return require;
00690 }
00691 #endif
```

6.7 build/CMakeFiles/3.19.7/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD __cplusplus

Functions

- int main (int argc, char *argv[])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * info_language_dialect_default

6.7.1 Macro Definition Documentation

6.7.1.1 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

Definition at line 546 of file CMakeCXXCompilerId.cpp.

6.7.1.2 COMPILER_ID

```
#define COMPILER_ID ""
```

Definition at line 299 of file CMakeCXXCompilerId.cpp.

6.7.1.3 CXX_STD

```
#define CXX_STD __cplusplus
```

Definition at line 638 of file CMakeCXXCompilerId.cpp.

6.7.1.4 DEC

```
#define DEC(
              n )
```

Value:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

Definition at line 550 of file CMakeCXXCompilerId.cpp.

6.7.1.5 HEX

```
#define HEX(
              n )
```

Value:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

Definition at line 561 of file CMakeCXXCompilerId.cpp.

6.7.1.6 PLATFORM_ID

```
#define PLATFORM_ID
```

Definition at line 424 of file CMakeCXXCompilerId.cpp.

6.7.1.7 STRINGIFY

```
#define STRINGIFY(
              X ) STRINGIFY_HELPER(X)
```

Definition at line 320 of file CMakeCXXCompilerId.cpp.

6.7.1.8 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X ) #X
```

Definition at line 319 of file CMakeCXXCompilerId.cpp.

6.7.2 Function Documentation

6.7.2.1 main()

```
int main (
             int argc,
             char * argv[] )
```

Definition at line 657 of file CMakeCXXCompilerId.cpp.

6.7.3 Variable Documentation

6.7.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

Definition at line 621 of file CMakeCXXCompilerId.cpp.

6.7.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

Definition at line 306 of file CMakeCXXCompilerId.cpp.

6.7.3.3 info_language_dialect_default

```
const char* info_language_dialect_default
```

Initial value:

```
= "INFO" ":" "dialect_default["
  "98"
"]"
```

Definition at line 641 of file CMakeCXXCompilerId.cpp.

6.7.3.4 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

Definition at line 620 of file CMakeCXXCompilerId.cpp.

6.8 CMakeCXXCompilerId.cpp

```
00001 /* This source file must have a .cpp extension so that all C++ compilers
00002    recognize the extension without flags.  Borland does not know .cxx for
00003    example.  */
00004 #ifndef __cplusplus
00005 # error "A C compiler has been selected for C++."
00006 #endif
00007
00008
00009 /* Version number components: V=Version, R=Revision, P=Patch
00010    Version date components:   YYYY=Year, MM=Month,   DD=Day  */
00011
00012 #if defined(__COMO__)
00013 # define COMPILER_ID "Comeau"
00014   /* __COMO_VERSION__ = VRR */
00015 # define COMPILER_VERSION_MAJOR DEC(__COMO_VERSION__ / 100)
00016 # define COMPILER_VERSION_MINOR DEC(__COMO_VERSION__ % 100)
00017
00018 #elif defined(__INTEL_COMPILER) || defined(__ICC)
00019 # define COMPILER_ID "Intel"
00020 # if defined(_MSC_VER)
00021 #  define SIMULATE_ID "MSVC"
00022 # endif
00023 # if defined(__GNUC__)
00024 #  define SIMULATE_ID "GNU"
00025 # endif
00026   /* __INTEL_COMPILER = VRP */
00027 # define COMPILER_VERSION_MAJOR DEC(__INTEL_COMPILER/100)
00028 # define COMPILER_VERSION_MINOR DEC(__INTEL_COMPILER/10 % 10)
00029 # if defined(__INTEL_COMPILER_UPDATE)
00030 #  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER_UPDATE)
00031 # else
00032 #  define COMPILER_VERSION_PATCH DEC(__INTEL_COMPILER   % 10)
00033 # endif
00034 # if defined(__INTEL_COMPILER_BUILD_DATE)
00035   /* __INTEL_COMPILER_BUILD_DATE = YYYYMMDD */
00036 #  define COMPILER_VERSION_TWEAK DEC(__INTEL_COMPILER_BUILD_DATE)
00037 # endif
00038 # if defined(_MSC_VER)
00039   /* _MSC_VER = VVRR */
00040 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00041 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00042 # endif
00043 # if defined(__GNUC__)
00044 #  define SIMULATE_VERSION_MAJOR DEC(__GNUC__)
00045 # elif defined(__GNUG__)
00046 #  define SIMULATE_VERSION_MAJOR DEC(__GNUG__)
00047 # endif
00048 # if defined(__GNUC_MINOR__)
00049 #  define SIMULATE_VERSION_MINOR DEC(__GNUC_MINOR__)
00050 # endif
00051 # if defined(__GNUC_PATCHLEVEL__)
00052 #  define SIMULATE_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00053 # endif
00054
00055 #elif defined(__PATHCC__)
00056 # define COMPILER_ID "PathScale"
00057 # define COMPILER_VERSION_MAJOR DEC(__PATHCC__)
00058 # define COMPILER_VERSION_MINOR DEC(__PATHCC_MINOR__)
00059 # if defined(__PATHCC_PATCHLEVEL__)
00060 #  define COMPILER_VERSION_PATCH DEC(__PATHCC_PATCHLEVEL__)
00061 # endif
00062
00063 #elif defined(__BORLANDC__) && defined(__CODEGEARC_VERSION__)
```

```
00064 # define COMPILER_ID "Embarcadero"
00065 # define COMPILER_VERSION_MAJOR HEX(__CODEGEARC_VERSION__»24 & 0x00FF)
00066 # define COMPILER_VERSION_MINOR HEX(__CODEGEARC_VERSION__»16 & 0x00FF)
00067 # define COMPILER_VERSION_PATCH DEC(__CODEGEARC_VERSION__     & 0xFFFF)
00068
00069 #elif defined(__BORLANDC__)
00070 # define COMPILER_ID "Borland"
00071   /* __BORLANDC__ = 0xVRR */
00072 # define COMPILER_VERSION_MAJOR HEX(__BORLANDC__»8)
00073 # define COMPILER_VERSION_MINOR HEX(__BORLANDC__ & 0xFF)
00074
00075 #elif defined(__WATCOMC__) && __WATCOMC__ < 1200
00076 # define COMPILER_ID "Watcom"
00077    /* __WATCOMC__ = VVRR */
00078 # define COMPILER_VERSION_MAJOR DEC(__WATCOMC__ / 100)
00079 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00080 # if (__WATCOMC__ % 10) > 0
00081 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00082 # endif
00083
00084 #elif defined(__WATCOMC__)
00085 # define COMPILER_ID "OpenWatcom"
00086    /* __WATCOMC__ = VVRP + 1100 */
00087 # define COMPILER_VERSION_MAJOR DEC((__WATCOMC__ - 1100) / 100)
00088 # define COMPILER_VERSION_MINOR DEC((__WATCOMC__ / 10) % 10)
00089 # if (__WATCOMC__ % 10) > 0
00090 #  define COMPILER_VERSION_PATCH DEC(__WATCOMC__ % 10)
00091 # endif
00092
00093 #elif defined(__SUNPRO_CC)
00094 # define COMPILER_ID "SunPro"
00095 # if __SUNPRO_CC >= 0x5100
00096    /* __SUNPRO_CC = 0xVRRP */
00097 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»12)
00098 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xFF)
00099 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
00100 # else
00101    /* __SUNPRO_CC = 0xVRP */
00102 #  define COMPILER_VERSION_MAJOR HEX(__SUNPRO_CC»8)
00103 #  define COMPILER_VERSION_MINOR HEX(__SUNPRO_CC»4 & 0xF)
00104 #  define COMPILER_VERSION_PATCH HEX(__SUNPRO_CC    & 0xF)
00105 # endif
00106
00107 #elif defined(__HP_aCC)
00108 # define COMPILER_ID "HP"
00109   /* __HP_aCC = VVRRPP */
00110 # define COMPILER_VERSION_MAJOR DEC(__HP_aCC/10000)
00111 # define COMPILER_VERSION_MINOR DEC(__HP_aCC/100 % 100)
00112 # define COMPILER_VERSION_PATCH DEC(__HP_aCC     % 100)
00113
00114 #elif defined(__DECCXX)
00115 # define COMPILER_ID "Compaq"
00116   /* __DECCXX_VER = VVRRTPPPP */
00117 # define COMPILER_VERSION_MAJOR DEC(__DECCXX_VER/10000000)
00118 # define COMPILER_VERSION_MINOR DEC(__DECCXX_VER/100000  % 100)
00119 # define COMPILER_VERSION_PATCH DEC(__DECCXX_VER         % 10000)
00120
00121 #elif defined(__IBMCPP__) && defined(__COMPILER_VER__)
00122 # define COMPILER_ID "zOS"
00123   /* __IBMCPP__ = VRP */
00124 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00125 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00126 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)
00127
00128 #elif defined(__ibmxl__) && defined(__clang__)
00129 # define COMPILER_ID "XLClang"
00130 # define COMPILER_VERSION_MAJOR DEC(__ibmxl_version__)
00131 # define COMPILER_VERSION_MINOR DEC(__ibmxl_release__)
00132 # define COMPILER_VERSION_PATCH DEC(__ibmxl_modification__)
00133 # define COMPILER_VERSION_TWEAK DEC(__ibmxl_ptf_fix_level__)
00134
00135
00136 #elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ >= 800
00137 # define COMPILER_ID "XL"
00138   /* __IBMCPP__ = VRP */
00139 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00140 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00141 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)
00142
00143 #elif defined(__IBMCPP__) && !defined(__COMPILER_VER__) && __IBMCPP__ < 800
00144 # define COMPILER_ID "VisualAge"
00145   /* __IBMCPP__ = VRP */
00146 # define COMPILER_VERSION_MAJOR DEC(__IBMCPP__/100)
00147 # define COMPILER_VERSION_MINOR DEC(__IBMCPP__/10 % 10)
00148 # define COMPILER_VERSION_PATCH DEC(__IBMCPP__    % 10)
00149
00150 #elif defined(__PGI)
```

```
00151 # define COMPILER_ID "PGI"
00152 # define COMPILER_VERSION_MAJOR DEC(__PGIC__)
00153 # define COMPILER_VERSION_MINOR DEC(__PGIC_MINOR__)
00154 # if defined(__PGIC_PATCHLEVEL__)
00155 #  define COMPILER_VERSION_PATCH DEC(__PGIC_PATCHLEVEL__)
00156 # endif
00157
00158 #elif defined(_CRAYC)
00159 # define COMPILER_ID "Cray"
00160 # define COMPILER_VERSION_MAJOR DEC(_RELEASE_MAJOR)
00161 # define COMPILER_VERSION_MINOR DEC(_RELEASE_MINOR)
00162
00163 #elif defined(__TI_COMPILER_VERSION__)
00164 # define COMPILER_ID "TI"
00165   /* __TI_COMPILER_VERSION__ = VVVRRRPPP */
00166 # define COMPILER_VERSION_MAJOR DEC(__TI_COMPILER_VERSION__/1000000)
00167 # define COMPILER_VERSION_MINOR DEC(__TI_COMPILER_VERSION__/1000   % 1000)
00168 # define COMPILER_VERSION_PATCH DEC(__TI_COMPILER_VERSION__        % 1000)
00169
00170 #elif defined(__FUJITSU) || defined(__FCC_VERSION) || defined(__fcc_version)
00171 # define COMPILER_ID "Fujitsu"
00172
00173 #elif defined(__ghs__)
00174 # define COMPILER_ID "GHS"
00175 /* __GHS_VERSION_NUMBER = VVVVRP */
00176 # ifdef __GHS_VERSION_NUMBER
00177 # define COMPILER_VERSION_MAJOR DEC(__GHS_VERSION_NUMBER / 100)
00178 # define COMPILER_VERSION_MINOR DEC(__GHS_VERSION_NUMBER / 10 % 10)
00179 # define COMPILER_VERSION_PATCH DEC(__GHS_VERSION_NUMBER      % 10)
00180 # endif
00181
00182 #elif defined(__SCO_VERSION__)
00183 # define COMPILER_ID "SCO"
00184
00185 #elif defined(__ARMCC_VERSION) && !defined(__clang__)
00186 # define COMPILER_ID "ARMCC"
00187 #if __ARMCC_VERSION >= 1000000
00188   /* __ARMCC_VERSION = VRRPPPP */
00189   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/1000000)
00190   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 100)
00191   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION     % 10000)
00192 #else
00193   /* __ARMCC_VERSION = VRPPPP */
00194   # define COMPILER_VERSION_MAJOR DEC(__ARMCC_VERSION/100000)
00195   # define COMPILER_VERSION_MINOR DEC(__ARMCC_VERSION/10000 % 10)
00196   # define COMPILER_VERSION_PATCH DEC(__ARMCC_VERSION    % 10000)
00197 #endif
00198
00199
00200 #elif defined(__clang__) && defined(__apple_build_version__)
00201 # define COMPILER_ID "AppleClang"
00202 # if defined(_MSC_VER)
00203 #  define SIMULATE_ID "MSVC"
00204 # endif
00205 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00206 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00207 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00208 # if defined(_MSC_VER)
00209   /* _MSC_VER = VVRR */
00210 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00211 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00212 # endif
00213 # define COMPILER_VERSION_TWEAK DEC(__apple_build_version__)
00214
00215 #elif defined(__clang__) && defined(__ARMCOMPILER_VERSION)
00216 # define COMPILER_ID "ARMClang"
00217   # define COMPILER_VERSION_MAJOR DEC(__ARMCOMPILER_VERSION/1000000)
00218   # define COMPILER_VERSION_MINOR DEC(__ARMCOMPILER_VERSION/10000 % 100)
00219   # define COMPILER_VERSION_PATCH DEC(__ARMCOMPILER_VERSION     % 10000)
00220 # define COMPILER_VERSION_INTERNAL DEC(__ARMCOMPILER_VERSION)
00221
00222 #elif defined(__clang__)
00223 # define COMPILER_ID "Clang"
00224 # if defined(_MSC_VER)
00225 #  define SIMULATE_ID "MSVC"
00226 # endif
00227 # define COMPILER_VERSION_MAJOR DEC(__clang_major__)
00228 # define COMPILER_VERSION_MINOR DEC(__clang_minor__)
00229 # define COMPILER_VERSION_PATCH DEC(__clang_patchlevel__)
00230 # if defined(_MSC_VER)
00231   /* _MSC_VER = VVRR */
00232 #  define SIMULATE_VERSION_MAJOR DEC(_MSC_VER / 100)
00233 #  define SIMULATE_VERSION_MINOR DEC(_MSC_VER % 100)
00234 # endif
00235
00236 #elif defined(__GNUC__) || defined(__GNUG__)
00237 # define COMPILER_ID "GNU"
```

```
00238 # if defined(__GNUC__)
00239 #  define COMPILER_VERSION_MAJOR DEC(__GNUC__)
00240 # else
00241 #  define COMPILER_VERSION_MAJOR DEC(__GNUG__)
00242 # endif
00243 # if defined(__GNUC_MINOR__)
00244 #  define COMPILER_VERSION_MINOR DEC(__GNUC_MINOR__)
00245 # endif
00246 # if defined(__GNUC_PATCHLEVEL__)
00247 #  define COMPILER_VERSION_PATCH DEC(__GNUC_PATCHLEVEL__)
00248 # endif
00249
00250 #elif defined(_MSC_VER)
00251 # define COMPILER_ID "MSVC"
00252   /* _MSC_VER = VVRR */
00253 # define COMPILER_VERSION_MAJOR DEC(_MSC_VER / 100)
00254 # define COMPILER_VERSION_MINOR DEC(_MSC_VER % 100)
00255 # if defined(_MSC_FULL_VER)
00256 #  if _MSC_VER >= 1400
00257     /* _MSC_FULL_VER = VVRRPPPPP */
00258 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 100000)
00259 #  else
00260     /* _MSC_FULL_VER = VVRRPPPP */
00261 #   define COMPILER_VERSION_PATCH DEC(_MSC_FULL_VER % 10000)
00262 #  endif
00263 # endif
00264 # if defined(_MSC_BUILD)
00265 #  define COMPILER_VERSION_TWEAK DEC(_MSC_BUILD)
00266 # endif
00267
00268 #elif defined(__VISUALDSPVERSION__) || defined(__ADSPBLACKFIN__) || defined(__ADSPTS__) ||
      defined(__ADSP21000__)
00269 # define COMPILER_ID "ADSP"
00270 #if defined(__VISUALDSPVERSION__)
00271   /* __VISUALDSPVERSION__ = 0xVVRRPP00 */
00272 # define COMPILER_VERSION_MAJOR HEX(__VISUALDSPVERSION__»24)
00273 # define COMPILER_VERSION_MINOR HEX(__VISUALDSPVERSION__»16 & 0xFF)
00274 # define COMPILER_VERSION_PATCH HEX(__VISUALDSPVERSION__»8  & 0xFF)
00275 #endif
00276
00277 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00278 # define COMPILER_ID "IAR"
00279 # if defined(__VER__) && defined(__ICCARM__)
00280 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 1000000)
00281 #  define COMPILER_VERSION_MINOR DEC(((__VER__) / 1000) % 1000)
00282 #  define COMPILER_VERSION_PATCH DEC((__VER__) % 1000)
00283 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00284 # elif defined(__VER__) && (defined(__ICCAVR__) || defined(__ICCRX__) || defined(__ICCRH850__) ||
      defined(__ICCRL78__) || defined(__ICC430__) || defined(__ICCRISCV__) || defined(__ICCV850__) ||
      defined(__ICC8051__))
00285 #  define COMPILER_VERSION_MAJOR DEC((__VER__) / 100)
00286 #  define COMPILER_VERSION_MINOR DEC((__VER__) - (((__VER__) / 100)*100))
00287 #  define COMPILER_VERSION_PATCH DEC(__SUBVERSION__)
00288 #  define COMPILER_VERSION_INTERNAL DEC(__IAR_SYSTEMS_ICC__)
00289 # endif
00290
00291
00292 /* These compilers are either not known or too old to define an
00293   identification macro.  Try to identify the platform and guess that
00294   it is the native compiler.  */
00295 #elif defined(__hpux) || defined(__hpua)
00296 # define COMPILER_ID "HP"
00297
00298 #else /* unknown compiler */
00299 # define COMPILER_ID ""
00300 #endif
00301
00302 /* Construct the string literal in pieces to prevent the source from
00303    getting matched.  Store it in a pointer rather than an array
00304    because some compilers will just produce instructions to fill the
00305    array rather than assigning a pointer to a static array.  */
00306 char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]";
00307 #ifdef SIMULATE_ID
00308 char const* info_simulate = "INFO" ":" "simulate[" SIMULATE_ID "]";
00309 #endif
00310
00311 #ifdef __QNXNTO__
00312 char const* qnxnto = "INFO" ":" "qnxnto[]";
00313 #endif
00314
00315 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00316 char const *info_cray = "INFO" ":" "compiler_wrapper[CrayPrgEnv]";
00317 #endif
00318
00319 #define STRINGIFY_HELPER(X) #X
00320 #define STRINGIFY(X) STRINGIFY_HELPER(X)
00321
```

```
00322 /* Identify known platforms by name.  */
00323 #if defined(__linux) || defined(__linux__) || defined(linux)
00324 # define PLATFORM_ID "Linux"
00325
00326 #elif defined(__CYGWIN__)
00327 # define PLATFORM_ID "Cygwin"
00328
00329 #elif defined(__MINGW32__)
00330 # define PLATFORM_ID "MinGW"
00331
00332 #elif defined(__APPLE__)
00333 # define PLATFORM_ID "Darwin"
00334
00335 #elif defined(_WIN32) || defined(__WIN32__) || defined(WIN32)
00336 # define PLATFORM_ID "Windows"
00337
00338 #elif defined(__FreeBSD__) || defined(__FreeBSD)
00339 # define PLATFORM_ID "FreeBSD"
00340
00341 #elif defined(__NetBSD__) || defined(__NetBSD)
00342 # define PLATFORM_ID "NetBSD"
00343
00344 #elif defined(__OpenBSD__) || defined(__OPENBSD)
00345 # define PLATFORM_ID "OpenBSD"
00346
00347 #elif defined(__sun) || defined(sun)
00348 # define PLATFORM_ID "SunOS"
00349
00350 #elif defined(_AIX) || defined(__AIX) || defined(__AIX__) || defined(__aix) || defined(__aix__)
00351 # define PLATFORM_ID "AIX"
00352
00353 #elif defined(__hpux) || defined(__hpux__)
00354 # define PLATFORM_ID "HP-UX"
00355
00356 #elif defined(__HAIKU__)
00357 # define PLATFORM_ID "Haiku"
00358
00359 #elif defined(__BeOS) || defined(__BEOS__) || defined(_BEOS)
00360 # define PLATFORM_ID "BeOS"
00361
00362 #elif defined(__QNX__) || defined(__QNXNTO__)
00363 # define PLATFORM_ID "QNX"
00364
00365 #elif defined(__tru64) || defined(_tru64) || defined(__TRU64__)
00366 # define PLATFORM_ID "Tru64"
00367
00368 #elif defined(__riscos) || defined(__riscos__)
00369 # define PLATFORM_ID "RISCos"
00370
00371 #elif defined(__sinix) || defined(__sinix__) || defined(__SINIX__)
00372 # define PLATFORM_ID "SINIX"
00373
00374 #elif defined(__UNIX_SV__)
00375 # define PLATFORM_ID "UNIX_SV"
00376
00377 #elif defined(__bsdos__)
00378 # define PLATFORM_ID "BSDOS"
00379
00380 #elif defined(_MPRAS) || defined(MPRAS)
00381 # define PLATFORM_ID "MP-RAS"
00382
00383 #elif defined(__osf) || defined(__osf__)
00384 # define PLATFORM_ID "OSF1"
00385
00386 #elif defined(_SCO_SV) || defined(SCO_SV) || defined(sco_sv)
00387 # define PLATFORM_ID "SCO_SV"
00388
00389 #elif defined(__ultrix) || defined(__ultrix__) || defined(_ULTRIX)
00390 # define PLATFORM_ID "ULTRIX"
00391
00392 #elif defined(__XENIX__) || defined(_XENIX) || defined(XENIX)
00393 # define PLATFORM_ID "Xenix"
00394
00395 #elif defined(__WATCOMC__)
00396 # if defined(__LINUX__)
00397 #  define PLATFORM_ID "Linux"
00398
00399 # elif defined(__DOS__)
00400 #  define PLATFORM_ID "DOS"
00401
00402 # elif defined(__OS2__)
00403 #  define PLATFORM_ID "OS2"
00404
00405 # elif defined(__WINDOWS__)
00406 #  define PLATFORM_ID "Windows3x"
00407
00408 # elif defined(__VXWORKS__)
```

```
00409 #  define PLATFORM_ID "VxWorks"
00410
00411 # else /* unknown platform */
00412 #  define PLATFORM_ID
00413 # endif
00414
00415 #elif defined(__INTEGRITY)
00416 # if defined(INT_178B)
00417 #  define PLATFORM_ID "Integrity178"
00418
00419 # else /* regular Integrity */
00420 #  define PLATFORM_ID "Integrity"
00421 # endif
00422
00423 #else /* unknown platform */
00424 # define PLATFORM_ID
00425
00426 #endif
00427
00428 /* For windows compilers MSVC and Intel we can determine
00429    the architecture of the compiler being used.  This is because
00430    the compilers do not have flags that can change the architecture,
00431    but rather depend on which compiler is being used
00432 */
00433 #if defined(_WIN32) && defined(_MSC_VER)
00434 # if defined(_M_IA64)
00435 #  define ARCHITECTURE_ID "IA64"
00436
00437 # elif defined(_M_X64) || defined(_M_AMD64)
00438 #  define ARCHITECTURE_ID "x64"
00439
00440 # elif defined(_M_IX86)
00441 #  define ARCHITECTURE_ID "X86"
00442
00443 # elif defined(_M_ARM64)
00444 #  define ARCHITECTURE_ID "ARM64"
00445
00446 # elif defined(_M_ARM)
00447 #  if _M_ARM == 4
00448 #   define ARCHITECTURE_ID "ARMV4I"
00449 #  elif _M_ARM == 5
00450 #   define ARCHITECTURE_ID "ARMV5I"
00451 #  else
00452 #   define ARCHITECTURE_ID "ARMV" STRINGIFY(_M_ARM)
00453 #  endif
00454
00455 # elif defined(_M_MIPS)
00456 #  define ARCHITECTURE_ID "MIPS"
00457
00458 # elif defined(_M_SH)
00459 #  define ARCHITECTURE_ID "SHx"
00460
00461 # else /* unknown architecture */
00462 #  define ARCHITECTURE_ID ""
00463 # endif
00464
00465 #elif defined(__WATCOMC__)
00466 # if defined(_M_I86)
00467 #  define ARCHITECTURE_ID "I86"
00468
00469 # elif defined(_M_IX86)
00470 #  define ARCHITECTURE_ID "X86"
00471
00472 # else /* unknown architecture */
00473 #  define ARCHITECTURE_ID ""
00474 # endif
00475
00476 #elif defined(__IAR_SYSTEMS_ICC__) || defined(__IAR_SYSTEMS_ICC)
00477 # if defined(__ICCARM__)
00478 #  define ARCHITECTURE_ID "ARM"
00479
00480 # elif defined(__ICCRX__)
00481 #  define ARCHITECTURE_ID "RX"
00482
00483 # elif defined(__ICCRH850__)
00484 #  define ARCHITECTURE_ID "RH850"
00485
00486 # elif defined(__ICCRL78__)
00487 #  define ARCHITECTURE_ID "RL78"
00488
00489 # elif defined(__ICCRISCV__)
00490 #  define ARCHITECTURE_ID "RISCV"
00491
00492 # elif defined(__ICCAVR__)
00493 #  define ARCHITECTURE_ID "AVR"
00494
00495 # elif defined(__ICC430__)
```

```
00496 #  define ARCHITECTURE_ID "MSP430"
00497
00498 # elif defined(__ICCV850__)
00499 #  define ARCHITECTURE_ID "V850"
00500
00501 # elif defined(__ICC8051__)
00502 #  define ARCHITECTURE_ID "8051"
00503
00504 # else /* unknown architecture */
00505 #  define ARCHITECTURE_ID ""
00506 # endif
00507
00508 #elif defined(__ghs__)
00509 # if defined(__PPC64__)
00510 #  define ARCHITECTURE_ID "PPC64"
00511
00512 # elif defined(__ppc__)
00513 #  define ARCHITECTURE_ID "PPC"
00514
00515 # elif defined(__ARM__)
00516 #  define ARCHITECTURE_ID "ARM"
00517
00518 # elif defined(__x86_64__)
00519 #  define ARCHITECTURE_ID "x64"
00520
00521 # elif defined(__i386__)
00522 #  define ARCHITECTURE_ID "X86"
00523
00524 # else /* unknown architecture */
00525 #  define ARCHITECTURE_ID ""
00526 # endif
00527
00528 #elif defined(__TI_COMPILER_VERSION__)
00529 # if defined(__TI_ARM__)
00530 #  define ARCHITECTURE_ID "ARM"
00531
00532 # elif defined(__MSP430__)
00533 #  define ARCHITECTURE_ID "MSP430"
00534
00535 # elif defined(__TMS320C28XX__)
00536 #  define ARCHITECTURE_ID "TMS320C28x"
00537
00538 # elif defined(__TMS320C6X__) || defined(_TMS320C6X)
00539 #  define ARCHITECTURE_ID "TMS320C6x"
00540
00541 # else /* unknown architecture */
00542 #  define ARCHITECTURE_ID ""
00543 # endif
00544
00545 #else
00546 #  define ARCHITECTURE_ID
00547 #endif
00548
00549 /* Convert integer to decimal digit literals.  */
00550 #define DEC(n)                   \
00551   ('0' + (((n) / 10000000)%10)), \
00552   ('0' + (((n) / 1000000)%10)),  \
00553   ('0' + (((n) / 100000)%10)),   \
00554   ('0' + (((n) / 10000)%10)),    \
00555   ('0' + (((n) / 1000)%10)),     \
00556   ('0' + (((n) / 100)%10)),      \
00557   ('0' + (((n) / 10)%10)),       \
00558   ('0' +  ((n) % 10))
00559
00560 /* Convert integer to hex digit literals.  */
00561 #define HEX(n)             \
00562   ('0' + ((n)>>28 & 0xF)), \
00563   ('0' + ((n)>>24 & 0xF)), \
00564   ('0' + ((n)>>20 & 0xF)), \
00565   ('0' + ((n)>>16 & 0xF)), \
00566   ('0' + ((n)>>12 & 0xF)), \
00567   ('0' + ((n)>>8  & 0xF)), \
00568   ('0' + ((n)>>4  & 0xF)), \
00569   ('0' + ((n)     & 0xF))
00570
00571 /* Construct a string literal encoding the version number components. */
00572 #ifdef COMPILER_VERSION_MAJOR
00573 char const info_version[] = {
00574   'I', 'N', 'F', 'O', ':',
00575   'c','o','m','p','i','l','e','r','_','v','e','r','s','i','o','n','[',
00576   COMPILER_VERSION_MAJOR,
00577 # ifdef COMPILER_VERSION_MINOR
00578   '.', COMPILER_VERSION_MINOR,
00579 #  ifdef COMPILER_VERSION_PATCH
00580    '.', COMPILER_VERSION_PATCH,
00581 #   ifdef COMPILER_VERSION_TWEAK
00582     '.', COMPILER_VERSION_TWEAK,
```

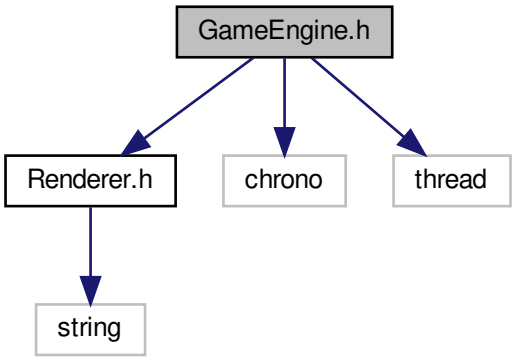
```
00583 #   endif
00584 #  endif
00585 # endif
00586   ’]’,’\0’};
00587 #endif
00588
00589 /* Construct a string literal encoding the internal version number. */
00590 #ifdef COMPILER_VERSION_INTERNAL
00591 char const info_version_internal[] = {
00592   ’I’, ’N’, ’F’, ’O’, ’:’,
00593   ’c’,’o’,’m’,’p’,’i’,’l’,’e’,’r’,’_’,’v’,’e’,’r’,’s’,’i’,’o’,’n’,’_’,
00594   ’i’,’n’,’t’,’e’,’r’,’n’,’a’,’l’,’[’,
00595   COMPILER_VERSION_INTERNAL,’]’,’\0’};
00596 #endif
00597
00598 /* Construct a string literal encoding the version number components. */
00599 #ifdef SIMULATE_VERSION_MAJOR
00600 char const info_simulate_version[] = {
00601   ’I’, ’N’, ’F’, ’O’, ’:’,
00602   ’s’,’i’,’m’,’u’,’l’,’a’,’t’,’e’,’_’,’v’,’e’,’r’,’s’,’i’,’o’,’n’,’[’,
00603   SIMULATE_VERSION_MAJOR,
00604 # ifdef SIMULATE_VERSION_MINOR
00605   ’.’, SIMULATE_VERSION_MINOR,
00606 #  ifdef SIMULATE_VERSION_PATCH
00607   ’.’, SIMULATE_VERSION_PATCH,
00608 #   ifdef SIMULATE_VERSION_TWEAK
00609   ’.’, SIMULATE_VERSION_TWEAK,
00610 #   endif
00611 #  endif
00612 # endif
00613   ’]’,’\0’};
00614 #endif
00615
00616 /* Construct the string literal in pieces to prevent the source from
00617    getting matched.  Store it in a pointer rather than an array
00618    because some compilers will just produce instructions to fill the
00619    array rather than assigning a pointer to a static array.  */
00620 char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]";
00621 char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]";
00622
00623
00624
00625 #if defined(__INTEL_COMPILER) && defined(_MSVC_LANG) && _MSVC_LANG < 201403L
00626 #  if defined(__INTEL_CXX11_MODE__)
00627 #    if defined(__cpp_aggregate_nsdmi)
00628 #      define CXX_STD 201402L
00629 #    else
00630 #      define CXX_STD 201103L
00631 #    endif
00632 #  else
00633 #    define CXX_STD 199711L
00634 #  endif
00635 #elif defined(_MSC_VER) && defined(_MSVC_LANG)
00636 #  define CXX_STD _MSVC_LANG
00637 #else
00638 #  define CXX_STD __cplusplus
00639 #endif
00640
00641 const char* info_language_dialect_default = "INFO" ":" "dialect_default["
00642 #if CXX_STD > 201703L
00643   "20"
00644 #elif CXX_STD >= 201703L
00645   "17"
00646 #elif CXX_STD >= 201402L
00647   "14"
00648 #elif CXX_STD >= 201103L
00649   "11"
00650 #else
00651   "98"
00652 #endif
00653 "]";
00654
00655 /*--------------------------------------------------------------------------*/
00656
00657 int main(int argc, char* argv[])
00658 {
00659   int require = 0;
00660   require += info_compiler[argc];
00661   require += info_platform[argc];
00662 #ifdef COMPILER_VERSION_MAJOR
00663   require += info_version[argc];
00664 #endif
00665 #ifdef COMPILER_VERSION_INTERNAL
00666   require += info_version_internal[argc];
00667 #endif
00668 #ifdef SIMULATE_ID
00669   require += info_simulate[argc];
```

```
00670 #endif
00671 #ifdef SIMULATE_VERSION_MAJOR
00672   require += info_simulate_version[argc];
00673 #endif
00674 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
00675   require += info_cray[argc];
00676 #endif
00677   require += info_language_dialect_default[argc];
00678   (void)argv;
00679   return require;
00680 }
```

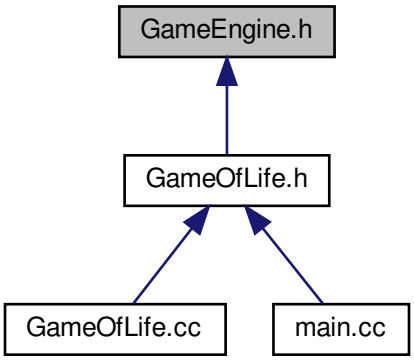
6.9 GameEngine.h File Reference

```
#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.

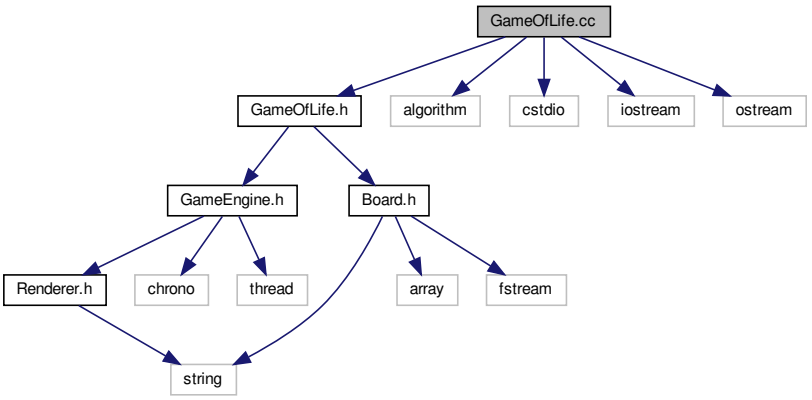
6.10 GameEngine.h

```
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
```

6.11 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)

6.11.1 Function Documentation

6.11.1.1 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.

6.12 GameOfLife.cc

```
00001 //
00002 // Created by mateu on 3/24/2021.
00003 //
00004
```

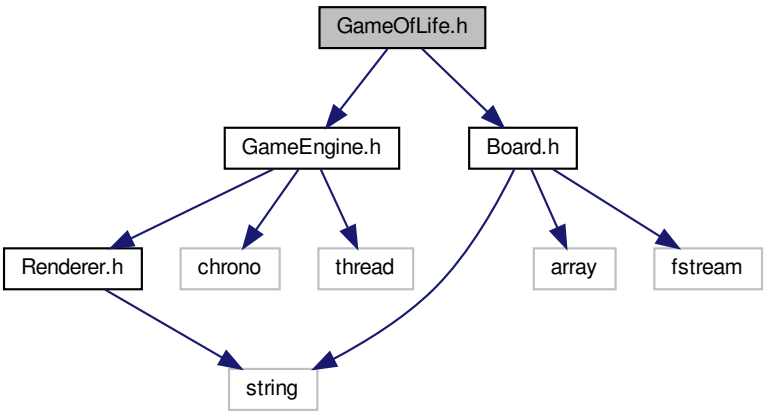
```
00005 #include "GameOfLife.h"
00006 #include <algorithm>
00007 #include <cstdio>
00008 #include <iostream>
00009 #include <ostream>
00010 #include <cstdio>
00011
00012 static void show_board(const Board &board) {
00013   for (int i = 0; i < board.size(); i++) {
00014     std::cout « board(i) « " ";
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 }
```

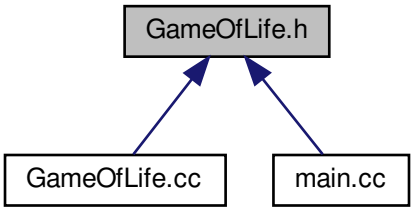
6.13 GameOfLife.h File Reference

```
#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)

6.13.1 Function Documentation

6.13.1.1 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.

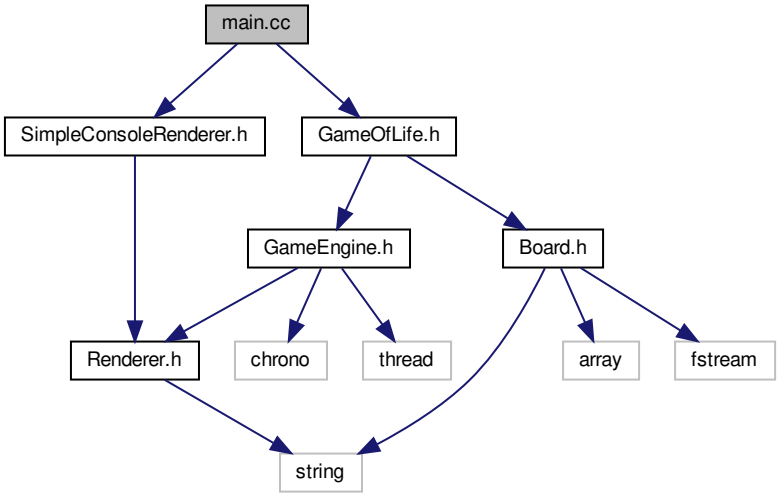
6.14 GameOfLife.h

```
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
```

6.15 main.cc File Reference

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

Include dependency graph for main.cc:



Functions

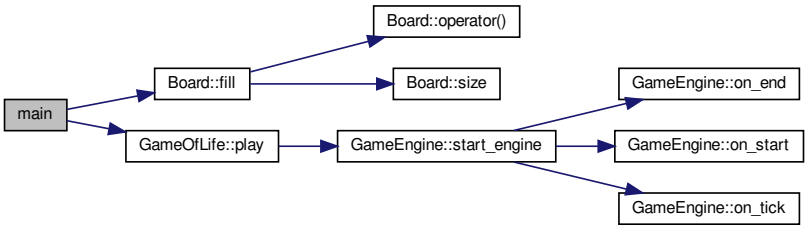
- int main ()

6.15.1 Function Documentation

6.15.1.1 main()

```
int main ( )
```

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



6.16 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
```

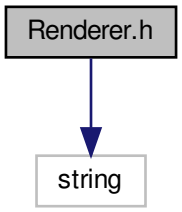
```
00008     GameEngine::Config config{};
00009
00010     config.framerate = 1;
00011     config.renderer = new SimpleConsoleRenderer;
00012
00013     Board board(size_x, size_y);
00014
00015     // Make the board empty
00016     board.fill(false);
00017
00018     // Setup some config on the board
00019     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 }
```

6.17 README.md File Reference

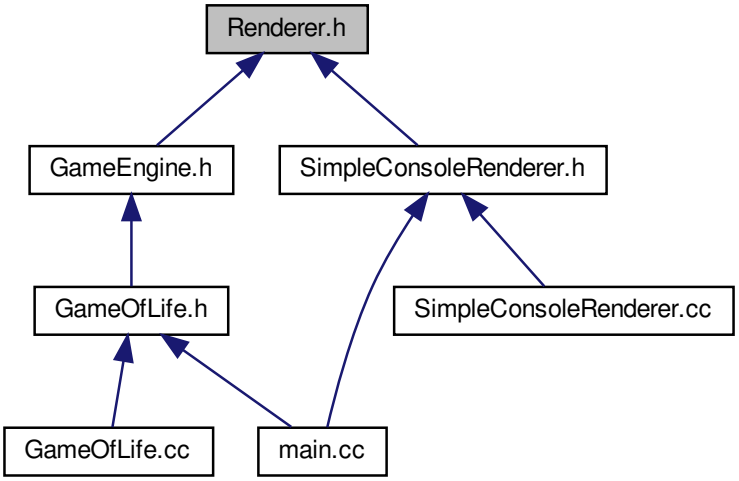
6.18 Renderer.h File Reference

`#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.

6.18.1 Enumeration Type Documentation

6.18.1.1 Color

`enum Color [strong]`

Colors used in renderer.

Enumerator

Red	
Green	
Blue	
Black	
White	

Definition at line 20 of file Renderer.h.

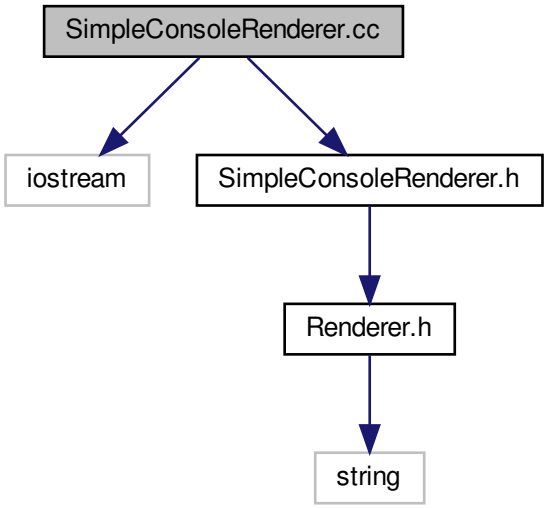
6.19 Renderer.h

```
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
```

6.20 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

6.20.1 Macro Definition Documentation

6.20.1.1 COLOR_BLACK

```
#define COLOR_BLACK 301
```

Definition at line 16 of file SimpleConsoleRenderer.cc.

6.20.1.2 COLOR_WHITE

```
#define COLOR_WHITE 300
```

Definition at line 17 of file SimpleConsoleRenderer.cc.

6.20.1.3 IS_ALPHA_NUMERIC

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

Definition at line 15 of file SimpleConsoleRenderer.cc.

6.21 SimpleConsoleRenderer.cc

```
00001 //
00002 // Created by mateu on 4/11/2021.
```

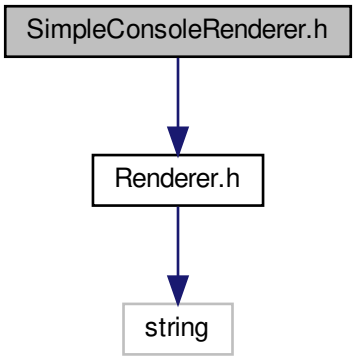
```
00003 //
00004
00005 #include <iostream>
00006 #include "SimpleConsoleRenderer.h"
00007
00008
00009 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32) && !defined(__CYGWIN__)
00010
00011 #include <Windows.h>
00012
00013 #endif
00014
00015 #define IS_ALPHA_NUMERIC(x) (x < 256)
00016 #define 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 }
```

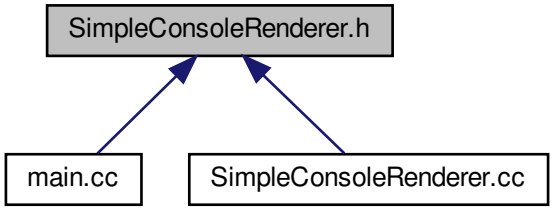
6.22 SimpleConsoleRenderer.h File Reference

`#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

6.22.1 Typedef Documentation

6.22.1.1 GrayscalePixel

```
typedef uint16_t GrayscalePixel
```

Definition at line 10 of file SimpleConsoleRenderer.h.

6.23 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