Game Of Life

Generated by Doxygen 1.8.11

Contents

Chapter 1

Game of Life - Version 1.0

General info

"The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. The "game" is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves." - Wikipedia

Demo

There are some demo files included in this project. See the demo directory for these files and start with parameter -f <filename> to test them out.

How to run "gameoflife"

Example: gameoflife.exe -er pontus -or daniel -g 100 -s 50x50

- -h help
- -er <even rulename> [default = conway]
- -or <odd rulename> [default = conway]
- -g <generations> [default = 500]
- -s <widthxheight> [default = 80x24]
- -f <filename> [default = random state]

Rules

- conway
  - Any live cell with fewer than two live neighbours dies, as if caused by under-population.
  - Any live cell with two or three live neighbours lives on to the next generation.
  - Any live cell with more than three live neighbours dies, as if by over-population.
  - Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- pontus
  - A living cell with one to seven neighbours will live on
  - A living cell with zero or eight neighbours will die
  - A dead cell with one to seven neighbours will live
  - A dead cell with zero or eight neighbours will continue to be dead
- gustav
  - A dead cell is of color RED
  - If a dead cell has exactly two living neighbours it will revive
  - If a living cell has any living neighbours that are exactly age 2, it will die
  - If a cell has 6 to 8 neighbours that are alive, it will die
  - If a dead cell has one living neighbours that is above age 0, it will revive
- daniel
  - If a living cell is older then age 4 it dies.
  - If a living cell has 4 or more neighbours it dies.
  - If a dead cell has 2 or more neighbours where at least 2 of them is age 1 or more. The dead cell will live.
  - If cell age 1: COLOR YELLOW.
  - If cell age 2: COLOR CYAN.
  - If cell age 3: COLOR BLUE.
  - If cell age 4: COLOR RED.

File format

First you need to specify the size of the board by width $*$ height. Then you need to specify all the cells that should be alive in the initialize.

An example is presented below
20x20
10,10
10,11
11,40
10,39

Developers

Pontus Stenlund
Daniel Jennebo
Gustav Olsson

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

Class Documentation

4.1 Cell Class Reference

This class represents a cell that is a 1 by 1 tile in a cellmap. The cell can be either dead or alive and has an age. It also has one color for alive and one for dead.

`#include <Cell.h>`

Public Member Functions

- Cell ()

  Default constructor for Cell class. Sets colorAlive & colorDead color to defaults.
- ~Cell ()

  Default destructor for Cell class, no implementation.
- void kill ()

  Function kills a cell. It sets the alive to false and age to 0.
- void revive ()

  Function revives a cell. It sets the alive to true.
- bool isAlive () const

  Function returns the state of cell (alive or dead).
- COLOR getColorAlive () const

  Function gets a cells alive color.
- void setColorAlive (COLOR pColor)

  Function sets the color for when the cell is alive.
- COLOR getColorDead () const

  Function gets a cells dead color.
- void setColorDead (COLOR pColor)

  Function sets the color for when the cell is dead.
- int getAge () const

  Function gets a cells age.
- void setAge (int pAge)

  Function sets age on a cell.

Private Attributes

- COLOR colorAlive
- COLOR colorDead
- bool alive = false
- int age = 0

4.1.1 Detailed Description

This class represents a cell that is a 1 by 1 tile in a cellmap. The cell can be either dead or alive and has an age. It also has one color for alive and one for dead.

Author

Daniel Jennebo

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Cell::Cell ( )

Default constructor for Cell class. Sets colorAlive & colorDead color to defaults.

Author

Daniel Jennebo

4.1.2.2 Cell::~Cell ( )

Default destructor for Cell class, no implementation.

Author

Daniel Jennebo

4.1.3 Member Function Documentation

4.1.3.1 int Cell::getAge ( ) const

Function gets a cells age.

Author

Daniel Jennebo

Returns

Membervariable age

4.1.3.2 COLOR Cell::getColorAlive ( ) const

Function gets a cells alive color.

Author

Daniel Jennebo

Returns

Membervariable colorAlive

4.1.3.3 COLOR Cell::getColorDead ( ) const

Function gets a cells dead color.

Author

Daniel Jennebo

Returns

Membervariable colorDead

4.1.3.4 bool Cell::isAlive ( ) const

Function returns the state of cell (alive or dead).

Author

Daniel Jennebo

Returns

Membervariable alive.

4.1.3.5 void Cell::kill ( )

Function kills a cell. It sets the alive to false and age to 0.

Author

Daniel Jennebo

4.1.3.6 void Cell::revive ( )

Function revives a cell. It sets the alive to true.

Author

Daniel Jennebo

4.1.3.7 void Cell::setAge (int *pAge*)

Function sets age on a cell.

Author

Daniel Jennebo

Parameters

pAge	contains the value that membervariable age is going to set to.

4.1.3.8 void Cell::setColorAlive (COLOR *pColor*)

Function sets the color for when the cell is alive.

Author

Daniel Jennebo

Parameters

pColor	contains the color that membervariable colorAlive is going to set to.

4.1.3.9 void Cell::setColorDead (COLOR *pColor*)

Function sets the color for when the cell is dead.

Author

Daniel Jennebo

Parameters

pColor	contains the color that membervariable colorDead is going to set to.

4.1.4 Member Data Documentation

4.1.4.1 int Cell::age = 0 [private]

Membervariable that holds the age for cell

4.1.4.2 bool Cell::alive = false [private]

Membervariable that holds true or false if cell are alive or dead

4.1.4.3 COLOR Cell::colorAlive [private]

Membervariable that holds the color for alivecell

4.1.4.4 COLOR Cell::colorDead [private]

Membervariable that holds the color for deadcell

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

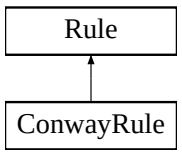
- C:/Users/Nollan/GitWork/gameoflife/include/Cell.h
- C:/Users/Nollan/GitWork/gameoflife/src/Cell.cpp

4.2 ConwayRule Class Reference

A concrete rule. Derived from base class Rule.

`#include <ConwayRule.h>`

Inheritance diagram for ConwayRule:



Public Member Functions

- void specificRule (Cell *cell, vector< Cell * > pNeighboursAlive)
  applies the specific rules for ConwayRule. Who lives and who dies.

4.2.1 Detailed Description

A concrete rule. Derived from base class Rule.

Author

Gustav Olsson

The rules are:

- A living cell that has fewer than two living neighbours will die
- A living cell with two to three living neighbours will survive
- A living cell with more than three living neighbours will die
- A dead cell with exactly three neighbours will reproduce

4.2.2 Member Function Documentation

4.2.2.1 void ConwayRule::specificRule (Cell * *cell,* vector< Cell * > *pNeighboursAlive*) [virtual]

applies the specific rules for ConwayRule. Who lives and who dies.

Author

Gustav Olsson

Parameters

cell	contains the current cell being checked in applyRules function.
pNeighboursAlive	contains the living cell neighbours to the current cell (param cell).

See also

Rule::applyRules(vector<vector<Cell*>> &cellMap)

Implements Rule.

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

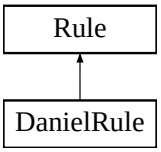
- C:/Users/Nollan/GitWork/gameoflife/include/ConwayRule.h
- C:/Users/Nollan/GitWork/gameoflife/src/ConwayRule.cpp

4.3 DanielRule Class Reference

A concrete rule. Derived from base class Rule.

`#include <DanielRule.h>`

Inheritance diagram for DanielRule:



Public Member Functions

- void specificRule (Cell *cell, vector< Cell * > pNeighboursAlive)
  applies the specific rules for DanielRule. Who lives and who dies.

4.3.1 Detailed Description

A concrete rule. Derived from base class Rule.

Author

Daniel Jennebo

The rules are:

- If a living cell is older then age 4 it dies.
- If a living cell has 4 or more neighbours it dies.
- If a dead cell has 2 or more neighbours where at least 2 of them is age 1 or more. The dead cell will live.
- Colors:
  - If cell age 1: COLOR YELLOW.
  - If cell age 2: COLOR CYAN.
  - If cell age 3: COLOR BLUE.
  - If cell age 4: COLOR RED.

4.3.2 Member Function Documentation

4.3.2.1 void DanielRule::specificRule (Cell * *cell*, vector< Cell * > *pNeighboursAlive*) [virtual]

applies the specific rules for DanielRule. Who lives and who dies.

Author

Daniel Jennebo

Parameters

cell	contains the current cell being checked in applyRules function.
pNeighboursAlive	contains the living cell neighbours to the current cell (param cell).

See also

Rule::applyRules(vector<vector<Cell*>> &cellMap)

Implements Rule.

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

- C:/Users/Nollan/GitWork/gameoflife/include/DanielRule.h
- C:/Users/Nollan/GitWork/gameoflife/src/DanielRule.cpp

4.4 GameEngine Class Reference

This class handles runtime of program.

`#include <GameEngine.h>`

Public Member Functions

- GameEngine ()
  Default constuct for GameEngine class.
- ~GameEngine ()
  Destructor for GameEngine class. Deletes pointers in evenCellMap, oddCellMap & pointers oddRule, evenRule.
- void run ()
  Contains the game loop and draws cellmaps in the terminal window.
- void setWindowSize (std::string size)
  Function sets x and y values to determine size of terminal window.
- void setGenerations (std::string pGenerations)
  Function sets number of generations to iterate trough. Sets the membervariable generations to params value.
- int getGenerations () const
  Function returns membervariable generations.
- void setOddRule (Rule *rule)

Function sets membervariable oddRule value to params value.

- void setEvenRule (Rule *rule)

  Function sets membervariable evenRule value to params value.
- void readStartCellsFromFile (std::string file)

  Reads from file and sets windowsize and which cells to be alive.
- void setStartCellsRandom ()

  Function make random number of cells alive at random places in evenCellMap This is the initial state.
- int getX () const

  Function returns membervariable x.
- void setX (int pX)

  Function sets membervariable x to params value.
- int getY () const

  Function returns membervariable y.
- void setY (int pY)

  Function sets membervariable y to params value.
- std::string showHelp ()

  Function returns a string with help.
- std::vector< std::vector< Cell * > > getVector ()

  Returns the vector evenCellMap.
- void initCellMaps ()

  Function initiate the evenCellMap & oddCellMap vectors with dead cells.
- void drawOnScreen (vector< vector< Cell * >> pCellMap)

  This function handles draw on screen (print the alive and dead cells to terminal)

Private Attributes

- Rule * oddRule
- Rule * evenRule
- int x = 80
- int y = 24
- int generations = 500
- std::vector< std::vector< Cell * > > evenCellMap
- std::vector< std::vector< Cell * > > oddCellMap

4.4.1 Detailed Description

This class handles runtime of program.

Author

: Daniel Jennebo, Gustav Olsson, Pontus Stenlund

4.4.2 Constructor & Destructor Documentation

4.4.2.1 GameEngine::GameEngine ( )

Default constuct for GameEngine class.

Author

Daniel Jennebo.

4.4.2.2 GameEngine::~GameEngine ( )

Destructor for GameEngine class. Deletes pointers in evenCellMap, oddCellMap & pointers oddRule, evenRule.

Author

Daniel Jennebo.

4.4.3 Member Function Documentation

4.4.3.1 void GameEngine::drawOnScreen (vector< vector< Cell * >> *pCellMap*)

This function handles draw on screen (print the alive and dead cells to terminal)

Author

Daniel Jennebo, Gustav Olsson

4.4.3.2 int GameEngine::getGenerations ( ) const

Function returns membervariable generations.

Author

Daniel Jennebo.

Returns

membervariable generations as int.

4.4.3.3 std::vector< std::vector< Cell * > > GameEngine::getVector ( )

Returns the vector evenCellMap.

Author

Daniel Jennebo.

4.4.3.4 int GameEngine::getX ( ) const

Function returns membervariable x.

Author

Daniel Jennebo.

Returns

membervariable x as int.

4.4.3.5 int GameEngine::getY ( ) const

Function returns membervariable y.

Author

Daniel Jennebo.

Returns

membervariable y as int.

4.4.3.6 void GameEngine::initCellMaps ( )

Function initiate the evenCellMap & oddCellMap vectors with dead cells.

Author

Daniel Jennebo.

4.4.3.7 void GameEngine::readStartCellsFromFile (std::string *file*)

Reads from file and sets windowsize and which cells to be alive.

Author

Pontus Stenlund

Parameters

file	is the string that holds the name of the file to be read.

4.4.3.8 void GameEngine::setEvenRule (Rule * *rule*)

Function sets membervariable evenRule value to params value.

Author

Daniel Jennebo.

Parameters

rule	contains a rule object.

4.4.3.9 void GameEngine::setGenerations (std::string *pGenerations*)

Function sets number of generations to iterate trough. Sets the membervariable generations to params value.

Author

Daniel Jennebo.

Parameters

pGenerations	contains a string with number of generations.

4.4.3.10 void GameEngine::setOddRule (Rule * *rule*)

Function sets membervariable oddRule value to params value.

Author

Daniel Jennebo.

Parameters

rule	contains a rule object.

4.4.3.11 void GameEngine::setStartCellsRandom ( )

Function make random number of cells alive at random places in evenCellMap This is the initial state.

Author

Daniel Jennebo.

4.4.3.12 void GameEngine::setWindowSize (std::string *size*)

Function sets x and y values to determine size of terminal window.

Author

Daniel Jennebo.

Parameters

size	contains a string with "WIDTHxHEIGHT".

4.4.3.13 void GameEngine::setX (int *pX*)

Function sets membervariable x to params value.

Author

Daniel Jennebo.

Parameters

pX	(int) contains number of columns (width).

4.4.3.14 void GameEngine::setY (int *pY*)

Function sets membervariable y to params value.

Author

Daniel Jennebo.

Parameters

pY	(int) contains number of rows (height).

4.4.3.15 std::string GameEngine::showHelp ( )

Function returns a string with help.

Author

Daniel Jennebo.

Returns

help - Contains help information.

4.4.4 Member Data Documentation

4.4.4.1 std::vector<std::vector<Cell*> > GameEngine::evenCellMap [private]

Member variable that holds the evenCellMap

4.4.4.2 Rule* GameEngine::evenRule [private]

Member variable that holds the evenRule

4.4.4.3 int GameEngine::generations = 500 [private]

Member variale that holds the generations value (the times to iterate)

4.4.4.4 std::vector<std::vector<Cell*> > GameEngine::oddCellMap [private]

Member variable that holds the oddCellMap

4.4.4.5 Rule* GameEngine::oddRule [private]

Member variable that holds the oddRule

4.4.4.6 int GameEngine::x = 80 [private]

Member variable that holds the width value of window

4.4.4.7 int GameEngine::y = 24 [private]

Member variable that holds the height value of window

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

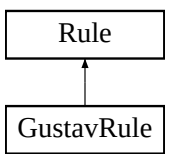
- C:/Users/Nollan/GitWork/gameoflife/include/GameEngine.h
- C:/Users/Nollan/GitWork/gameoflife/src/GameEngine.cpp

4.5 GustavRule Class Reference

A concrete rule. Derived from base class Rule.

`#include <GustavRule.h>`

Inheritance diagram for GustavRule:



Public Member Functions

- void specificRule (Cell *cell, vector< Cell * > pNeighboursAlive)

  applies the specific rules for GustavRule. Who lives and who dies.

4.5.1 Detailed Description

A concrete rule. Derived from base class Rule.

Author

Gustav Olsson

The rules are:

- A dead cell is of color RED
- If a dead cell has exactly two living neighbours it will revive
- If a living cell has any living neighbours that are exactly age 2, it will die
- If a cell has 6 to 8 neighbours that are alive, it will die
- If a dead cell has one living neighbours that is above age 0, it will revive

4.5.2 Member Function Documentation

4.5.2.1 void GustavRule::specificRule (Cell * *cell,* vector< Cell * > *pNeighboursAlive*) [virtual]

applies the specific rules for GustavRule. Who lives and who dies.

Author

Gustav Olsson

Parameters

cell	contains the current cell being checked in applyRules function.
pNeighboursAlive	contains the living cell neighbours to the current cell (param cell).

See also

Rule::applyRules(vector<vector<Cell*>> &cellMap)

Implements Rule.

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

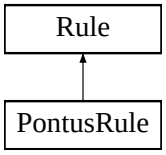
- C:/Users/Nollan/GitWork/gameoflife/include/GustavRule.h
- C:/Users/Nollan/GitWork/gameoflife/src/GustavRule.cpp

4.6 PontusRule Class Reference

A concrete rule. Derived from base class Rule.

`#include <PontusRule.h>`

Inheritance diagram for PontusRule:



Public Member Functions

- void specificRule (Cell *cell, vector< Cell * > pNeighboursAlive)
  Applies the specific rule for pontusRule of who lives and who dies.

4.6.1 Detailed Description

A concrete rule. Derived from base class Rule.

Author

Pontus Stenlund

the rules are:

- A living cell with one to seven neighbours will live on
- A living cell with zero or eight neighbours will die
- A dead cell with one to seven neighbours will live
- A dead cell with zero or eight neighbours will continue to be dead

4.6.2 Member Function Documentation

4.6.2.1 void PontusRule::specificRule (Cell * *cell,* vector< Cell * > *pNeighboursAlive*) [virtual]

Applies the specific rule for pontusRule of who lives and who dies.

Author

Pontus Stenlund

Parameters

cell	contains the current cell being checked in applyRules function.
pNeighboursAlive	contains the living cell neighbours to the current cell (param cell).

See also

Rule::applyRules(vector<vector<Cell*>> &cellMap)

Implements Rule.

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

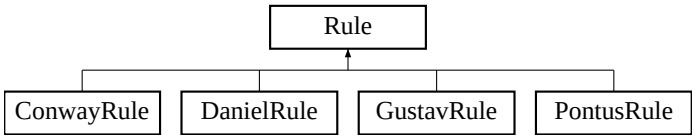
- C:/Users/Nollan/GitWork/gameoflife/include/PontusRule.h
- C:/Users/Nollan/GitWork/gameoflife/src/PontusRule.cpp

4.7 Rule Interface Reference

This is the base class for concrete implementations of rules.

`#include <Rule.h>`

Inheritance diagram for Rule:



Public Member Functions

- Rule ()

  Default construct, no implementation.
- virtual ~Rule ()

  Default destruct, no implementation.
- virtual void applyRules (const vector< vector< Cell * >> &cellMap, vector< vector< Cell * >> &newCell↩ Map)

  Counts neighbouring cells and applies rules to every cell. Invokes specificRule() to apply derived implementations of rules on cells.
- virtual void **specificRule** (Cell *cell, vector< Cell * > pNeighboursAlive)=0

4.7.1 Detailed Description

This is the base class for concrete implementations of rules.

Author

Gustav Olsson

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Rule::Rule ( ) [inline]

Default construct, no implementation.

Author

Gustav Olsson

4.7.2.2 virtual Rule::~Rule ( ) [inline], [virtual]

Default destruct, no implementation.

Author

Gustav Olsson

4.7.3 Member Function Documentation

4.7.3.1 void Rule::applyRules (const vector< vector< Cell * >> & *cellMap,* vector< vector< Cell * >> & *newCellMap*) [virtual]

Counts neighbouring cells and applies rules to every cell. Invokes specificRule() to apply derived implementations of rules on cells.

Author

Gustav Olsson

Parameters

cellMap	is a vector that contains the current generation of cells.
newCellMap	is a vector that will hold the next generation of cells, i.e. the cells after rules are applied.

See also

Rule::specificRule(Cell* cell, vector<Cell*> pNeighboursAlive)

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

- C:/Users/Nollan/GitWork/gameoflife/include/Rule.h
- C:/Users/Nollan/GitWork/gameoflife/src/Rule.cpp

4.8 RuleFactory Class Reference

This class is a factory that instantiate a rule based on user input.

`#include <RuleFactory.h>`

Public Member Functions

- RuleFactory ()

  Default construct, no implementation.

Static Public Member Functions

- static Rule * createRule (std::string rule)

  Returns an instance of a rule based on parameters value.

4.8.1 Detailed Description

This class is a factory that instantiate a rule based on user input.

Author

Daniel Jennebo

4.8.2 Constructor & Destructor Documentation

4.8.2.1 RuleFactory::RuleFactory ( )

Default construct, no implementation.

Author

Daniel Jennebo.

4.8.3 Member Function Documentation

4.8.3.1 Rule * RuleFactory::createRule (std::string *rule*) [static]

Returns an instance of a rule based on parameters value.

Author

Daniel Jennebo.

Parameters

rule	contains the name of the rule that should be created.

Returns

An instance of the derived rule.

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

- C:/Users/Nollan/GitWork/gameoflife/include/RuleFactory.h
- C:/Users/Nollan/GitWork/gameoflife/src/RuleFactory.cpp

4.9 Screen Class Reference

Public Member Functions

- **Screen** (uint16_t width, uint16_t height)
- void **clear** ()
- void **fill** (char ch, const TerminalColor &color)
- void **fillRect** (uint16_t x, uint16_t y, uint16_t w, uint16_t h, char ch, const TerminalColor &color)
- void **set** (uint16_t x, uint16_t y, char ch, const TerminalColor &color)
- void **setText** (uint16_t x, uint16_t y, const std::string &text, const TerminalColor &color)
- void **setTextRect** (uint16_t x, uint16_t y, uint16_t w, uint16_t h, const std::string &text, const TerminalColor &color)
- void **draw** (Terminal &terminal)

Private Attributes

- const uint16_t **m_width**
- const uint16_t **m_height**
- const uint32_t **m_size**
- TerminalColor * **m_color**
- char * **m_data**

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

- C:/Users/Nollan/GitWork/gameoflife/include/screen.h
- C:/Users/Nollan/GitWork/gameoflife/src/screen.cpp

4.10 Terminal Class Reference

Public Member Functions

- void **pushColor** (const TerminalColor &color)
- void **popColor** ()
- std::function< std::function< std::ostream &(std::ostream &)>const std::string &str)> **strColor** (const TerminalColor &color)
- std::function< std::ostream &(std::ostream &)> **color** (const TerminalColor &color)
- void **clearColors** ()
- void **clear** ()
- void **resetCursor** ()
- void **setCursor** (unsigned int x, unsigned int y)
- std::function< std::function< std::ostream &(std::ostream &)>unsigned int x, unsigned int y)> **position** ()
- void **showCursor** (bool show)

Private Member Functions

- void **setColor** (const TerminalColor &color)
- int **cti** (COLOR c)

Private Attributes

- std::stack< TerminalColor > **m_colors**

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

- C:/Users/Nollan/GitWork/gameoflife/terminal/terminal.h
- C:/Users/Nollan/GitWork/gameoflife/terminal/terminal.cpp

4.11 TerminalColor Class Reference

Public Member Functions

- **TerminalColor** (COLOR fg=COLOR::WHITE, COLOR bg=COLOR::BLACK)
- COLOR **fg** () const
- COLOR **bg** () const

Private Attributes

- COLOR **m_fg**
- COLOR **m_bg**

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

- C:/Users/Nollan/GitWork/gameoflife/terminal/terminal.h

Index