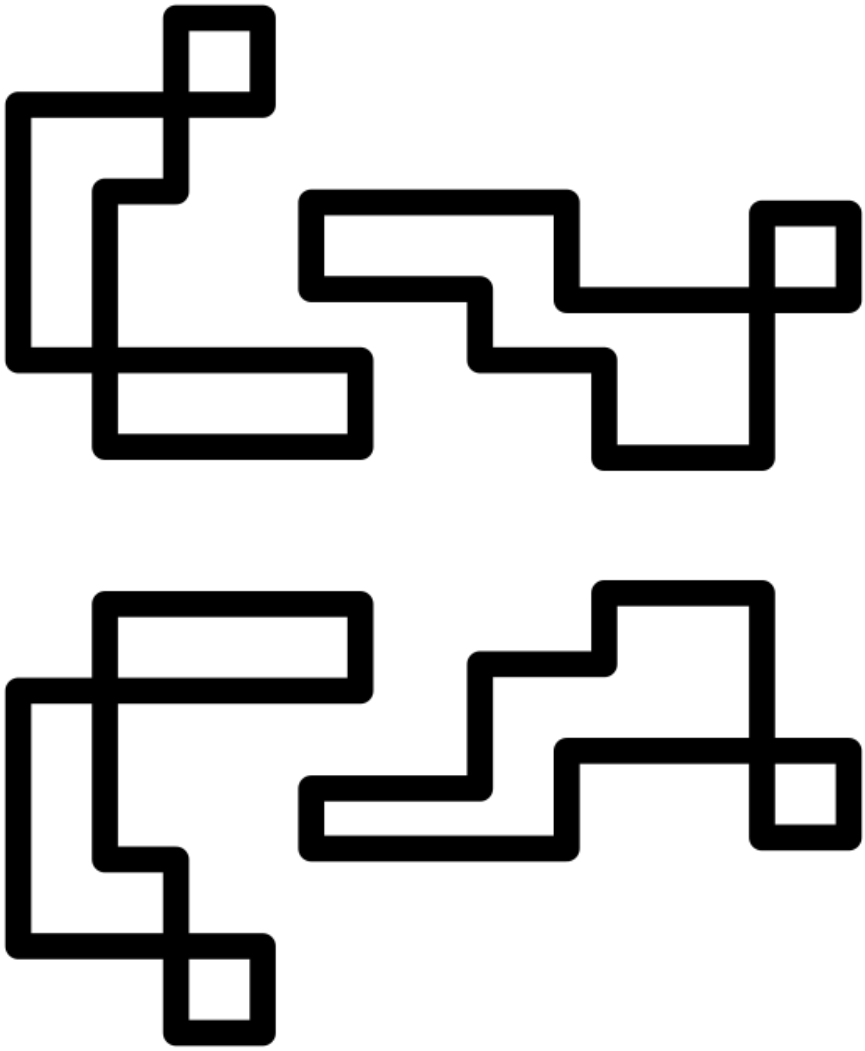

Conway's Game of Life Documentation

Release 1.0

Shehab and Darya

May 27, 2025

Contents

1 Modules **3**
- 1.1 Main Module . . . 3
- 1.2 Core Module . . . 3
- 1.3 GUI Module . . . 4

Python Module Index **8**

Index **9**

1 Modules

1.1 Main Module

This module initializes and starts the game.

1.2 Core Module

Submodules

game_of_life module

This module defines the core logic for customizable version of Conway's Game of Life. It provides grid state management and rules for updating generations.

Author: Shehabeldin Mohamed Version: 1.0

class core.game_of_life.**GameOfLife**(*width: int*, *height: int*, *wrap: bool = False*)

Bases: object

Supports wraparound edges and configurable rules as follows: A live cell survives if alive neighbors are within under/overpopulation limits. A dead cell becomes alive if it has exactly *custom_reproduction_number* neighbors.

Parameters

- **width** (*int*) – Width of the grid in cells.
- **height** (*int*) – Height of the grid in cells.
- **wrap** (*bool*) – Whether the grid wraps around the edges.

clear()

Reset the grid to all dead cells and reset generation count.

count_alive_neighbors(*x: int*, *y: int*) → int

Count the number of alive neighbors for the cell at (x, y).

Parameters

- **x** (*int*) – X-coordinate of the cell.
- **y** (*int*) – Y-coordinate of the cell.

Returns

(int) Number of alive neighboring cells.

get_generation() → int

Returns the current generation number.

next_generation()

Advance the simulation by one generation using standard Game of Life rules.

next_generation_custom()

Advance the simulation by one generation using custom rules.

reset_custom_rules()

Reset the rules to standard Game of Life rules

set_custom_rules(*overpop: int*, *underpop: int*, *repro: int*)

Set custom rules for cell survival and reproduction.

Parameters

- **overpop** (*int*) – Maximum neighbors before a cell dies from overpopulation.
- **underpop** (*int*) – Minimum neighbors for a live cell to survive.

- **repro** (*int*) – Exact number of neighbors required for a dead cell to reproduce.

`toggle_cell`(*x: int*, *y: int*)

Toggle the alive/dead state of a cell.

Parameters

- **x** (*int*) – X-coordinate of the cell.
- **y** (*int*) – Y-coordinate of the cell.

infinite_game module

`class` core.infinite_game.**InfiniteGameOfLife**

Bases: `object`

Implementation of Conway's Game of Life with infinite grid. Uses dictionary to store only live cells, allowing for infinite expansion. Coordinates can be any integer (positive or negative).

Author: Darya Sharnevich Version: 1.0

`_count_neighbors`(*x: int*, *y: int*) → int

Count live neighbors for a cell. Returns number of live neighbors

`_get_cells_to_check`()

Get set of all cells that need to be checked for the next generation.

`_update_bounds`()

Update the bounds of the live cells area.

`clear`()

Clear the grid and reset generation counter.

`next_generation`()

Calculate the next generation of cells.

`set_custom_rules`(*underpop: int*, *overpop: int*, *repro: int*)

Set custom rules for cell survival and reproduction.

`toggle_cell`(*x: int*, *y: int*)

Toggle cell state at given coordinates.

1.3 GUI Module

Main GUI Modules

game_gui module

Game of Life GUI using PyQt5

A zoomable, pannable, interactive GUI for Conway's Game of Life. Integrates both fixed and infinite grid implementations.

Author: Darya Sharnevich Version: 1.1

`class` gui.game_gui.**GameOfLifeGUI**(*menu_window=None*, *speed=10*, *fixed_view=False*, *width=None*, *height=None*, *wrap=False*)

Bases: `QWidget`

GUI for the Game of Life.

Parameters

- **menu_window** (*QWidget*) – Reference to menu window (optional).

- **speed** (*int*) – Initial simulation speed (generations / second).
- **fixed_view** (*bool*) – If True, grid has fixed width and height, panning/zoom is disabled.
- **width** (*int, optional*) – Width of the grid in cells (required for fixed grid mode).
- **height** (*int, optional*) – Height of the grid in cells (required for fixed grid mode).
- **wrap** (*bool, optional*) – Enable grid wrapping (only for fixed grid mode).

_start_timer_with_current_speed()

Start timer with current speed setting.

apply_dark_theme()

Apply dark theme colors and QSS.

apply_light_theme()

Apply light theme colors and QSS.

build_gui()

Build header, canvas, and controls layout.

change_speed()

Adjust simulation speed from slider.

clear_grid()

Clear grid and reset generation count.

confirm_exit_to_menu()

Shows confirmation dialog to return to the game_window menu.

next_generation()

Update the game_window state by one generation.

toggle_theme()

Switch between light and dark GUI themes.

toggle_timer()

Start or pause simulation timer.

start_menu module

class gui.start_menu.**MainMenu**

Bases: QWidget

Main menu window for the Game of Life application.

Provides options to start the game_window, open settings, view game_window info, or exit. Also handles game_window configuration including grid size, wrapping, speed, and custom rules.

_build_ui()

Build and arrange GUI elements in the main menu.

show_info()

Display information about Conway's Game of Life.

show_settings()

Display game_window settings dialog for configuration.

Allows users to configure: - Grid size (infinite/fixed) - Grid wrapping - Grid dimensions - Game speed - Custom game_window rules (survival and reproduction conditions)

start_game()

Start the game_window with the selected settings.

Game Modules Subpackage

control_panel module

class gui.game_modules.control_panel.**ControlPanel**(*start_callback*, *next_callback*, *clear_callback*, *theme_callback*, *speed_change_callback*, *initial_speed*)

Bases: QWidget

Control panel for game_window controls. Consists of: - Start/Pause button - Next generation button - Clear grid button - Theme toggle button - Speed control slider with labels

build_ui()

Build and arrange control panel UI elements with proper spacing and layout.

grid_canvas module

class gui.game_modules.grid_canvas.**GridCanvas**(*game*, *fixed_view_callable*, *zoom*, *offset*, *colors*, *parent=None*)

Bases: QWidget

Interactive canvas for displaying and manipulating the Game of Life grid.

Supports: - Cell toggling with left mouse button - Grid panning with right mouse button - Zoom with mouse wheel - Fixed and infinite grid modes - Custom color schemes

_draw_line_between_points(*x1*, *y1*, *x2*, *y2*)

Draw a continuous line of live cells between two points using Bresenham's algorithm.

_get_cell_coords(*pos*)

Convert screen coordinates to cell grid coordinates.

Parameters
pos – QPoint with screen coordinates

Returns
Tuple (x, y) with grid coordinates or None if outside grid

mouseMoveEvent(*event: QMouseEvent*)

Handle panning of the view when dragging and enables drawing between cells.

Right button: Pan the grid view Left button: Draw continuous line of live cells

mousePressEvent(*event: QMouseEvent*)

Handle cell toggling and drag start.

Left button: Toggle cell state Right button: Start grid panning

mouseReleaseEvent(*event*)

Reset drag state on mouse button release.

paintEvent(*event*)

Paint the grid and live cells based on current state.

wheelEvent(*event: QWheelEvent*)

Handle zoom in/out using mouse wheel. Maintains the center point during zoom.

header_bar module

class gui.game_modules.header_bar.**HeaderBar**(*parent=None*)

Bases: QWidget

Header bar widget for displaying game_window information.

build_ui()

Build and arrange header bar GUI elements.

set_generation(*gen_number*)

Update the generation counter display.

Python Module Index

c

core.game_of_life, 3
core.infinite_game, 4

g

gui.game_gui, 4
gui.game_modules.control_panel, 6
gui.game_modules.grid_canvas, 6
gui.game_modules.header_bar, 7
gui.start_menu, 5

m

main, 3

Index

Symbols

_build_ui() (*gui.start_menu.MainMenu method*), 5
_count_neighbors() (*core.infinite_game.InfiniteGameOfLife method*), 4
_draw_line_between_points() (*gui.game_modules.grid_canvas.GridCanvas method*), 6
_get_cell_coords() (*gui.game_modules.grid_canvas.GridCanvas method*), 6
_get_cells_to_check() (*core.infinite_game.InfiniteGameOfLife method*), 4
_start_timer_with_current_speed() (*gui.game_gui.GameOfLifeGUI method*), 5
_update_bounds() (*core.infinite_game.InfiniteGameOfLife method*), 4

A

apply_dark_theme() (*gui.game_gui.GameOfLifeGUI method*), 5
apply_light_theme() (*gui.game_gui.GameOfLifeGUI method*), 5

B

build_gui() (*gui.game_gui.GameOfLifeGUI method*), 5
build_ui() (*gui.game_modules.control_panel.ControlPanel method*), 6
build_ui() (*gui.game_modules.header_bar.HeaderBar method*), 7

C

change_speed() (*gui.game_gui.GameOfLifeGUI method*), 5
clear() (*core.game_of_life.GameOfLife method*), 3
clear() (*core.infinite_game.InfiniteGameOfLife method*), 4
clear_grid() (*gui.game_gui.GameOfLifeGUI method*), 5
confirm_exit_to_menu() (*gui.game_gui.GameOfLifeGUI method*), 5
ControlPanel (*class in gui.game_modules.control_panel*), 6
core.game_of_life
    module, 3
core.infinite_game
    module, 4
count_alive_neighbors() (*core.game_of_life.GameOfLife method*), 3

G

GameOfLife (*class in core.game_of_life*), 3
GameOfLifeGUI (*class in gui.game_gui*), 4
get_generation() (*core.game_of_life.GameOfLife method*), 3
GridCanvas (*class in gui.game_modules.grid_canvas*), 6
gui.game_gui
    module, 4
gui.game_modules.control_panel
    module, 6
gui.game_modules.grid_canvas
    module, 6
gui.game_modules.header_bar
    module, 7
gui.start_menu
    module, 5

H

HeaderBar (*class in gui.game_modules.header_bar*), 7

I

InfiniteGameOfLife (*class in core.infinite_game*), 4

M

main
    module, 3
MainMenu (*class in gui.start_menu*), 5
module
    core.game_of_life, 3
    core.infinite_game, 4
    gui.game_gui, 4
    gui.game_modules.control_panel, 6
    gui.game_modules.grid_canvas, 6
    gui.game_modules.header_bar, 7
    gui.start_menu, 5
    main, 3
mouseMoveEvent() (*gui.game_modules.grid_canvas.GridCanvas method*), 6
mousePressEvent() (*gui.game_modules.grid_canvas.GridCanvas method*), 6
mouseReleaseEvent() (*gui.game_modules.grid_canvas.GridCanvas method*), 6

N

next_generation() (*core.game_of_life.GameOfLife method*), 3

next_generation() (*core.infinite_game.InfiniteGameOfLife method*), 4
next_generation() (*gui.game_gui.GameOfLifeGUI method*), 5
next_generation_custom() (*core.game_of_life.GameOfLife method*), 3

P

paintEvent() (*gui.game_modules.grid_canvas.GridCanvas method*), 6

R

reset_custom_rules() (*core.game_of_life.GameOfLife method*), 3

S

set_custom_rules() (*core.game_of_life.GameOfLife method*), 3
set_custom_rules() (*core.infinite_game.InfiniteGameOfLife method*), 4
set_generation() (*gui.game_modules.header_bar.HeaderBar method*), 7
show_info() (*gui.start_menu.MainMenu method*), 5
show_settings() (*gui.start_menu.MainMenu method*), 5
start_game() (*gui.start_menu.MainMenu method*), 5

T

toggle_cell() (*core.game_of_life.GameOfLife method*), 4
toggle_cell() (*core.infinite_game.InfiniteGameOfLife method*), 4
toggle_theme() (*gui.game_gui.GameOfLifeGUI method*), 5
toggle_timer() (*gui.game_gui.GameOfLifeGUI method*), 5

W

wheelEvent() (*gui.game_modules.grid_canvas.GridCanvas method*), 6