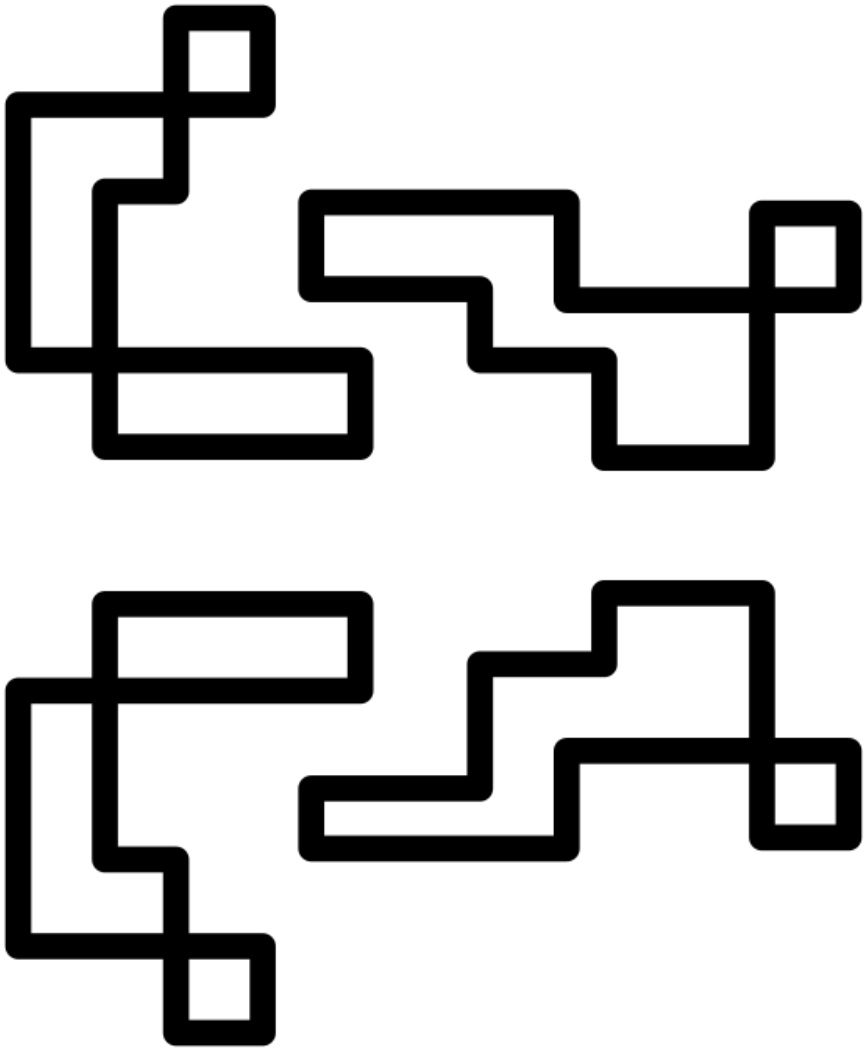

Conway's Game of Life Documentation

Release 1.0

Shehab and Darya

May 22, 2025

Contents

1 Modules

1.1 Main Module

This module initializes and starts the game.

1.2 Core Module

Game of Life Logic 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.

1.3 GUI Module

Game of Life GUI using PyQt5

This module provides a zoomable, pannable, interactive GUI for Conway's Game of Life, based on a GameOfLife class with methods like *toggle_cell(x, y)*, *next_generation()*, and *clear()*.

Author: Darya Sharnevich Version: 1.0

class gui.game_gui.**GameOfLifeGUI**(*width=50*, *height=50*, *wrap=False*, *menu_window=None*, *speed=10*, *fixed_view=False*)

Bases: QWidget

A PyQt5 widget for displaying and interacting with the Game of Life.

Parameters

- **width** (*int*) – Width of the grid in cells.
- **height** (*int*) – Height of the grid in cells.
- **wrap** (*bool*) – Enable grid wrapping behavior.
- **menu_window** (*QWidget*) – Optional reference to main menu window.
- **speed** (*int*) – Initial simulation speed (generations per second).
- **fixed_view** (*bool*) – Whether to disable zoom/pan and scale grid to fit.

_apply_dark_theme()

Apply dark theme colors and QSS.

_apply_light_theme()

Apply light theme colors and QSS.

_build_ui()

Initialize and arrange UI elements: header, canvas, controls.

_change_speed()

Adjust simulation speed from slider.

_clear_grid()

Clear grid and reset generation count.

_confirm_exit_to_menu()

Show confirmation dialog to return to main menu.

_get_cell_coords(*pos*)

Convert screen coordinates to cell grid coordinates.

_mouse_drag(*event: QMouseEvent*)

Handle panning of the view during dragging.

_mouse_press(*event: QMouseEvent*)

Handle cell toggling and drag start.

_mouse_release(*event: QMouseEvent*)

Reset drag state on release.

_mouse_wheel(*event: QWheelEvent*)

Zoom in or out using mouse wheel.

_next_generation()

Advance the game state by one generation.

_paint_grid(*event*)

Paint the grid and live cells based on current state.

_toggle_theme()

Switch between light and dark UI themes.

_toggle_timer()

Start or pause simulation timer.

Python Module Index

Index

Symbols

C

G

M

N

R

S

T