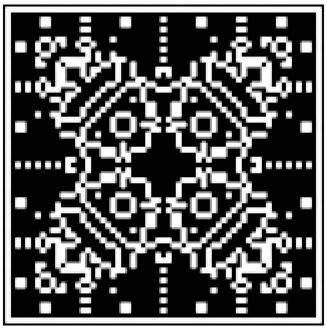


**Asset Name**: OmniCA
**Version**: 1.0.0
**Developer**: Game Soft Craft
**Contact Info**: gamesoftcraft@gmail.com

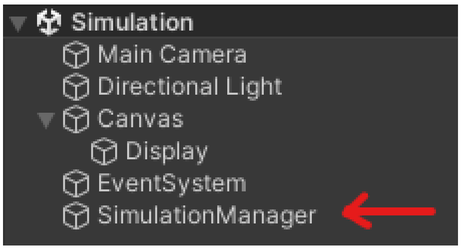
## Introduction:

OmniCA is an experimental Unity3D tool that simulates 2D, two-state cellular automata on a square grid. It uses a 9-cell neighborhood to generate rules from a massive 512-bit rule space—2^512 possibilities! With a GPU-accelerated engine, OmniCA runs efficiently and lets you easily create, save, and experiment with custom rules and initial conditions.

## Usage Guide:

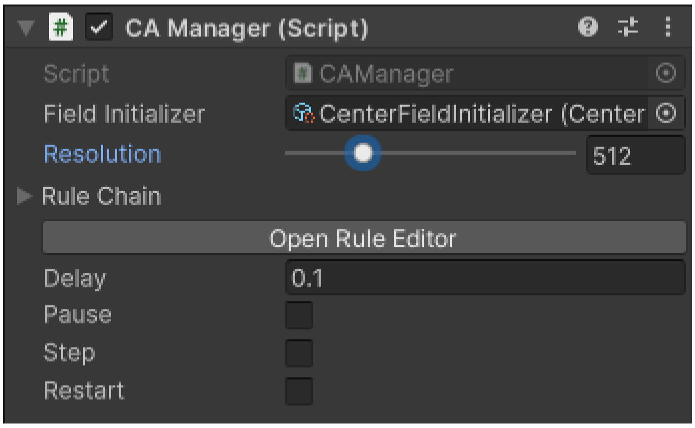
Follow these steps to use it:

- ➔ Open the Simulation Scene
- ➔ Locate the SimulationManager in the hierarchy.
- ➔ You can find it under Simulation > Canvas > SimulationManager in Simulation Hierarchy.



## SimulationManager Parameters:

The CAManager component allows you to control various aspects of the simulation:



CA Manager (Script)

Script: CAManager

Field Initializer: CenterFieldInitializer (Center

Resolution: 512

Rule Chain

Delay: 0.1

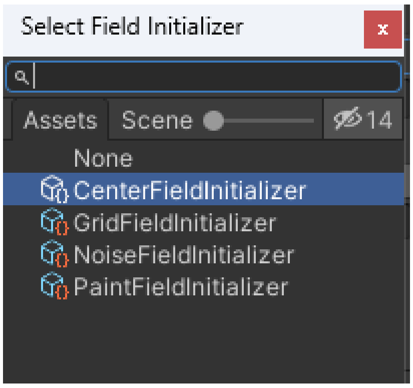
Pause: □

Step: □

Restart: □

### Field Initializer

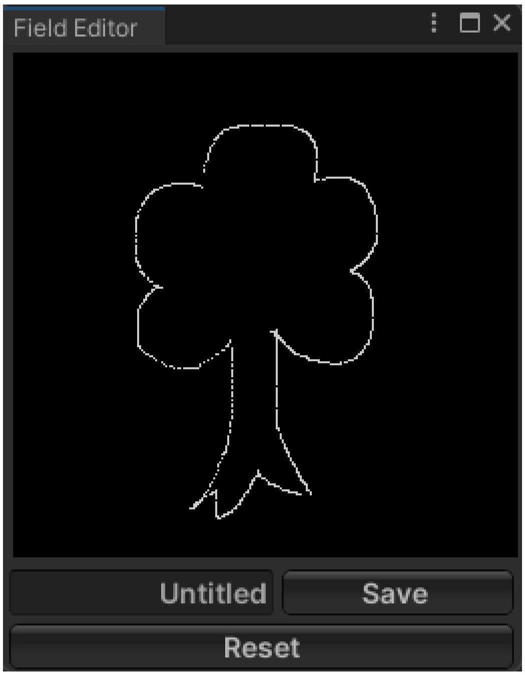
The field for a script object that sets the initial field configuration.



You can configure these or add new ones in the Assets/Data/FieldInitializers folder.

Standard options include:

- **Center**: doesn’t have any parameters, just initializing the central cell
- **Grid**: has *Area* and *Density* parameters, allowing initialization with different grids
- **Noise**: has *Density* parameter, controlling the noise on initialization
- **Paint**: Let you paint the initial field in a special window or use .png files in the "Texture" field.

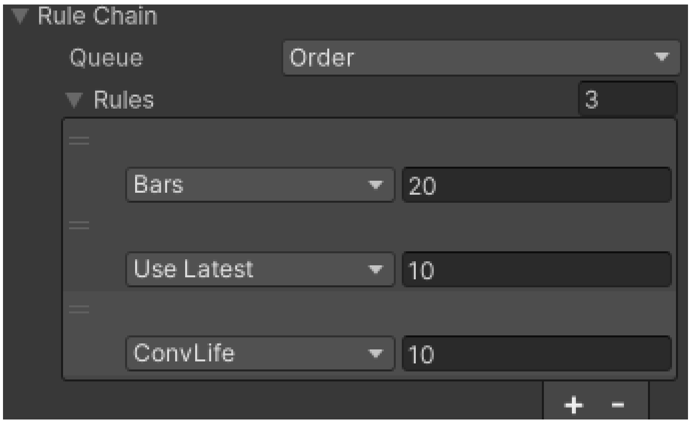


Create custom field initializers by creating classes derived from FieldInitializer. Refer to the existing ones for guidance.

## Resolution:

- Controls the field size. It is enforced to be a power of 2 for computational efficiency

## Rule Chain

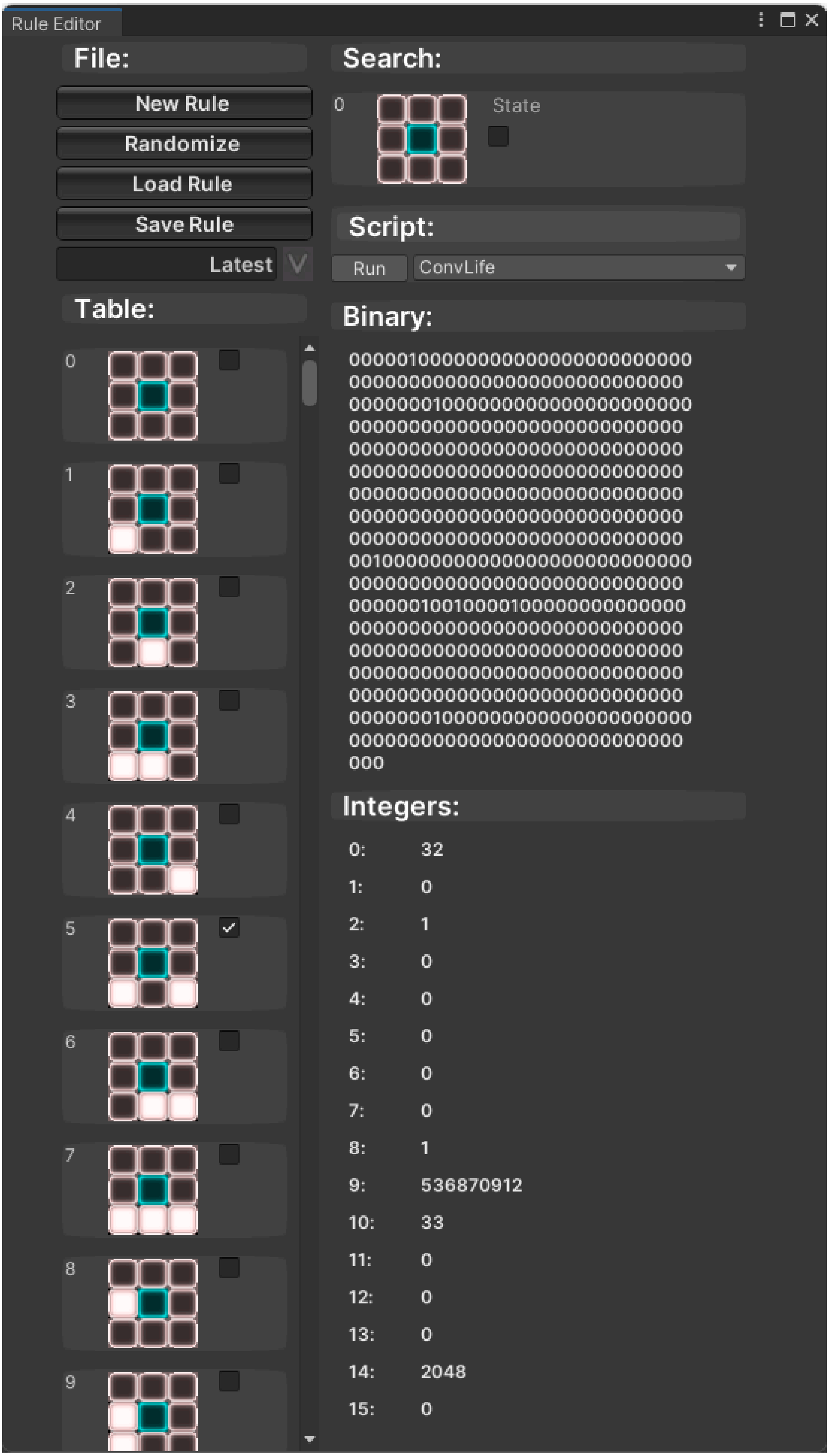


Rule Chain
Queue: Order
Rules: 3
Bars: 20
Use Latest: 10
ConvLife: 10

- Allows you to experiment with rule changes within a session.
- Queue Orders: Choose from Order, Repeat, or Random.
- The left side of the Rules element provides a dropdown list of saved rules, while the right side sets the number of steps before switching rules (0 for unlimited).
- Use Latest: Applies the most recent unsaved rule.

## Open Rule Editor

- Press the Open Rule Editor button to launch the Rule Editor window.



Binary:

```
00000100000000000000000000000
0000000000000000000000000000
00000001000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
00100000000000000000000000000
0000000000000000000000000000
00000010010000100000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
00000001000000000000000000000
0000000000000000000000000000
000
```

Integers:

```
0:     32
1:     0
2:     1
3:     0
4:     0
5:     0
6:     0
7:     0
8:     1
9:     536870912
10:    33
11:    0
12:    0
13:    0
14:    2048
15:    0
```

## Rule Editor Overview:

- **File:** Save, load, or create new rule files.
  It includes an input field and dropdown list of existing items, allowing you to edit rule names for saving.
- **Table:** The core of the rule editor, displaying all possible neighborhoods as grid-like icons.
  - A checkbox next to each icon indicates the next state of the cyan-highlighted cell. Setting or clearing the checkbox changes the rule when the displayed configuration is encountered.
- **Search:** Quickly access any neighborhood configuration.
  - Click on cells in the grid to toggle their state between filled and empty.
  - Once selected, you can modify the rule by setting/clearing the central cell's next state.
- **Script:** Generate rules via scripts.
  - Select a script from the dropdown menu and press Run to execute it.
  - Create custom rule-generating scripts by extending the RuleBuilder class. Refer to existing scripts as examples.
- **Binary:** Displays the current rule as bits, reflecting the states from the Table.
  - For example, if config #5 is set to "true" in the Table, it will show as "1" in the corresponding index in the Binary section.
  - Integers: Shows how the rule is stored in memory using 16 integers (512 bytes), representing the rule size.

## Other CAManager Fields:

- **Delay:** Controls the speed of the simulation. For example, 0.1 means a 0.1-second delay between frames (10 FPS).
- **Pause:** Pauses the simulation.
- **Step:** Allows for step-by-step simulation execution.
- **Restart**: Restarts the simulation from the beginning.