Game of Life Remastered ruleset

Tags	unime
Ultima Modifica	@July 21, 2021 6:48 PM

Similarities to the original *Life*

Like the original game of *Life*, this game can e played as a *zero-player game* in which only the board's initial state needs to set by the user. After the game starts, time moves in the form of discrete turns and all units obey to a fixed set of rules, so that there is no board state in which the simulation has to stop. The board is two-dimensional, all units fit into a single square and can interact with all neighboring units.

Units

As opposed to ordinary cells in cellular automatons, **Units** are what the game revolves around, and are comprised of the following properties:

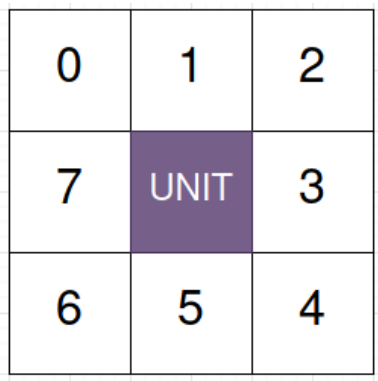
- **Health points**: Like in all video games, when an unit's HP drop to zero while the next turn is being computed, it will be removed from the board.
- **Species**: As the game is developed following OOP principles, each unit's Species determines their behavior, starting health points and rules for reproduction, population, hostility and additional actions.
- **State**: An unit's State applies variations to its behavior.
- **Rules**: The direct implementation of reproduction, population and newly added hostility rules in cellular automatons, which are also used to compute results of attacking units.
- **Independent Action**: If certain requirements are met, an unit may perform an additional action that only affects itself. These are yet to be used in practice.

Units provide a framework for different species to be implemented: this can be done by altering the initial values of the aforementioned properties on a particular Unit implementation:

- **LifeUnit**: Implements behavior for Life-like cellular automata, with the addition of hostile behavior.
- **Mimic**: Derived from LifeUnit, with the exception that it can attack and replicate units of a different species by changing its own rules.

Neighborhood

Moore's neighborhood is used to determine a unit's neighbors, which are passed to each unit when the game transitions to a new turn using an Array. The position in the array also defines the neighboring unit's *direction* relative to the processed unit. The following diagram illustrates the game's convention for creating the aforementioned arrays.



Computing next turn

Computation of the board state in the next turn is broken down into three steps:

Survival step

Executed on every non-empty square, this step sets a field that stores the unit's state in the simulation's next turn.

- Pass an array of neighboring units to the currently processed unit
- If population and/or hostility requirements are not met, the unit's HP will be decremented by 1.

- If the unit's HP didn't change up to this point, it's Independent Action will be executed.
- If the unit's own Independent Action has been executed, and the current state has its own Independent Action, it will be executed.
- If at this point the unit's HP are set to a value of 0 or less, the unit's next turn state will be set to dead.

Reproduction step

Executed on every empty square, this step calculates if a new unit should be born by taking into account the square's neighborhood and using the following rules.

- There can only be one newborn unit of a single species.
- The number of neighbors of a given species must meet the reproduction requirements for said species.
- If reproduction requirements are met for more than one species, the species with more units will be chosen
- If a tie occurs, the species higher on the food chain (*Species* enum ordinal) will be chosen.

Cleanup step

Every unit's current state is updated, and all dead units are removed from the board.